US006330639B1

(12) **United States Patent**
Fanning et al.

(10) **Patent No.: US 6,330,639 B1**
(45) **Date of Patent: Dec. 11, 2001**

(54) **METHOD AND APPARATUS FOR DYNAMICALLY CHANGING THE SIZES OF POOLS THAT CONTROL THE POWER CONSUMPTION LEVELS OF MEMORY DEVICES**

(75) Inventors: **Blaise B. Fanning**, Cameron Park; **Jeffrey R. Wilcox**; **Khong S. Foo**, both of Folsom, all of CA (US)

(73) Assignee: **Intel Corporation**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/342,347**

(22) Filed: **Jun. 29, 1999**

(51) **Int. Cl.**[7] **G06F 1/32**; G06F 12/02

(52) **U.S. Cl.** **711/106**; 711/105; 365/222; 365/226; 713/320

(58) **Field of Search** 711/100, 105, 711/106, 154; 365/222, 226, 227, 228, 229; 713/320, 323, 324, 330

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,710,903		12/1987	Hereth et al. .	
5,396,635	*	3/1995	Fung	713/323
5,404,543		4/1995	Faucher et al. .	
5,758,175	*	5/1998	Fung	713/323
5,781,782	*	7/1998	Tachikawa	713/330
5,799,198	*	8/1998	Fung	713/323
5,877,651	*	3/1999	Furutani	327/538
5,892,959	*	4/1999	Fung	713/323
5,901,103		5/1999	Harris, II et al. .	
5,928,365	*	7/1999	Yoshida	713/324
5,966,725	*	10/1999	Tabo	711/106
6,021,502	*	2/2000	Ando	713/340
6,079,025	*	6/2000	Fung	713/323
6,115,823	*	9/2000	Velasco et al.	713/322

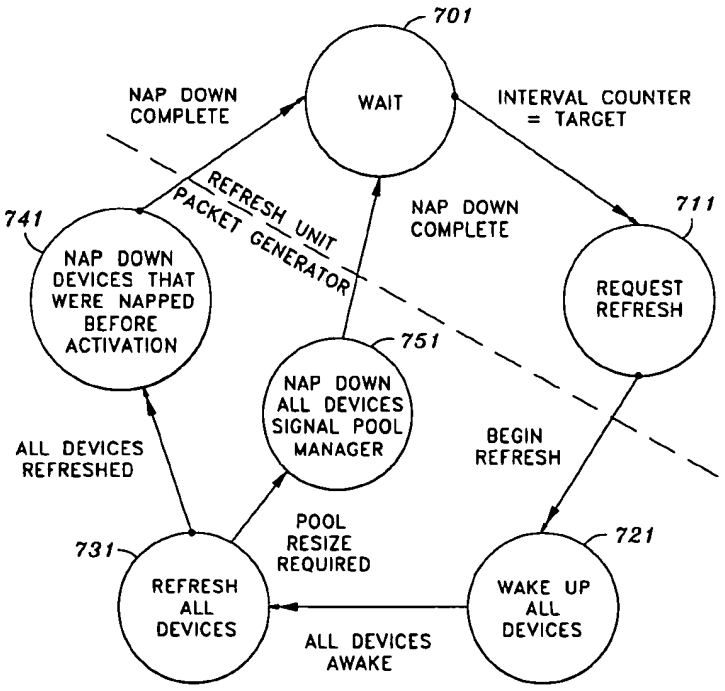
* cited by examiner

Primary Examiner—Tuan V. Thai
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

The present invention provides a method, apparatus, and system for dynamically changing the sizes of power-control pools that are used to control the power consumption levels of memory devices. In one embodiment, a request to change the sizes of the memory power-control pools is received. In response to receiving the request to change the sizes of the memory power-control pools, the memory devices are placed in a specific operating mode or power state after being refreshed in a periodic refresh cycle. In response to a signal indicating that all memory devices have been placed in the specific operating mode, powercontrol pools are resized according to pool size values corresponding to the request received.

49 Claims, 14 Drawing Sheets



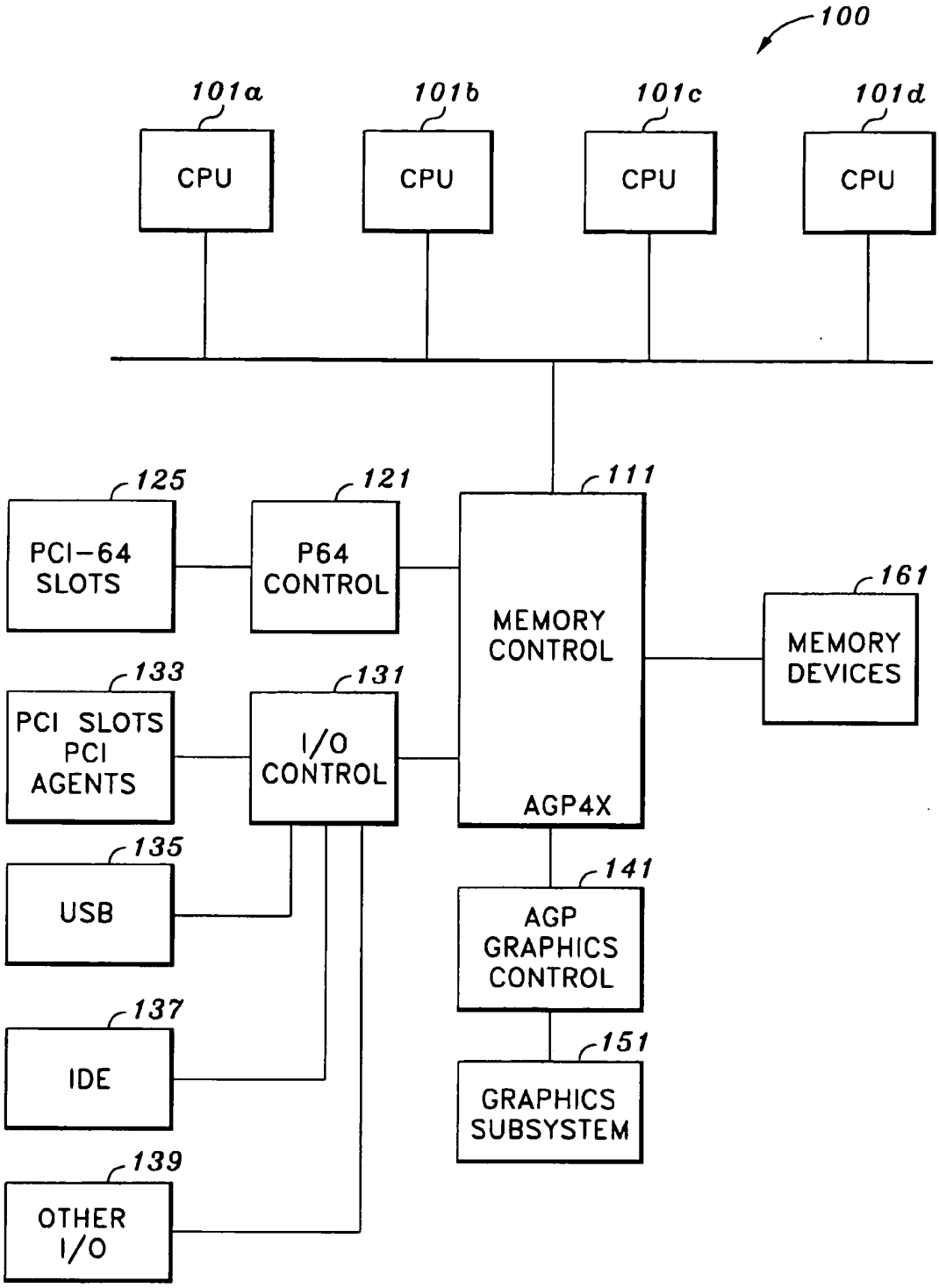

100
101a
CPU
101b
CPU
101c
CPU
101d
CPU
125
PCI–64 SLOTS
121
P64 CONTROL
111
MEMORY CONTROL
AGP4X
161
MEMORY DEVICES
133
PCI SLOTS PCI AGENTS
131
I/O CONTROL
135
USB
137
IDE
139
OTHER I/O
141
AGP GRAPHICS CONTROL
151
GRAPHICS SUBSYSTEM

FIG. 1

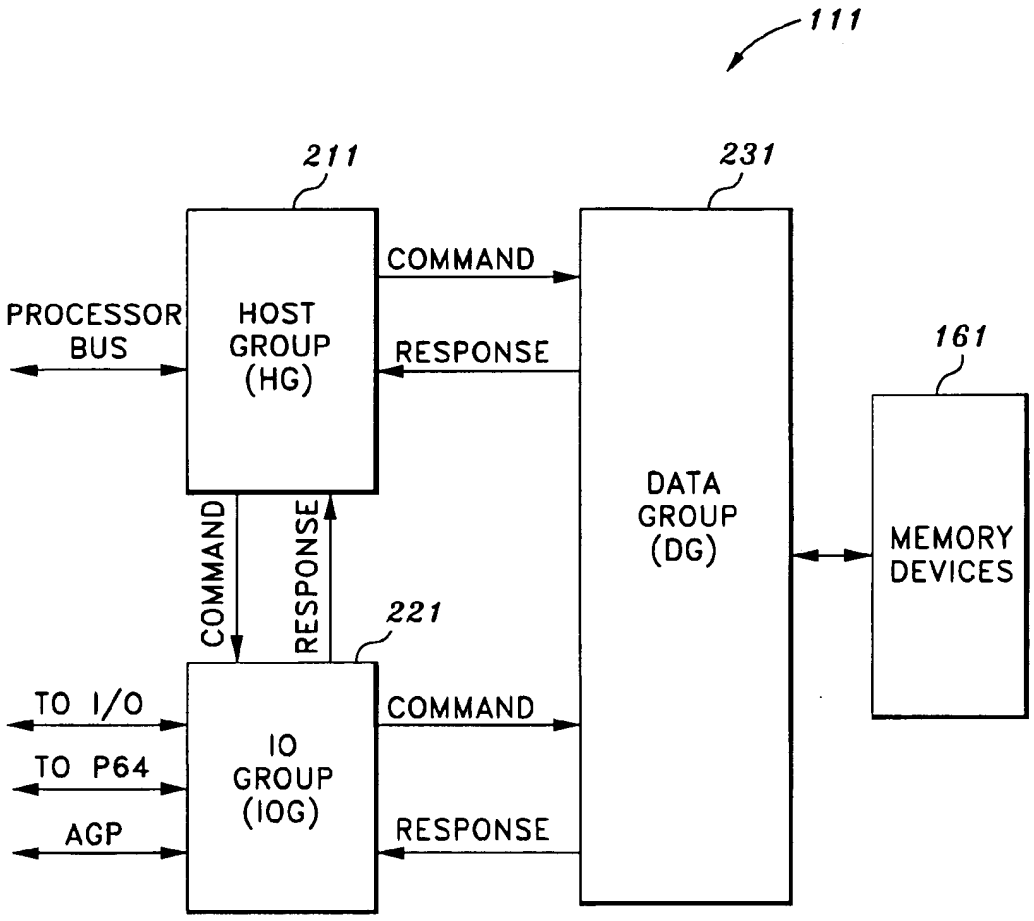

111
211
231
161
PROCESSOR BUS
HOST GROUP (HG)
COMMAND
RESPONSE
DATA GROUP (DG)
MEMORY DEVICES
COMMAND
RESPONSE
221
TO I/O
TO P64
AGP
IO GROUP (IOG)
COMMAND
RESPONSE

FIG. 2

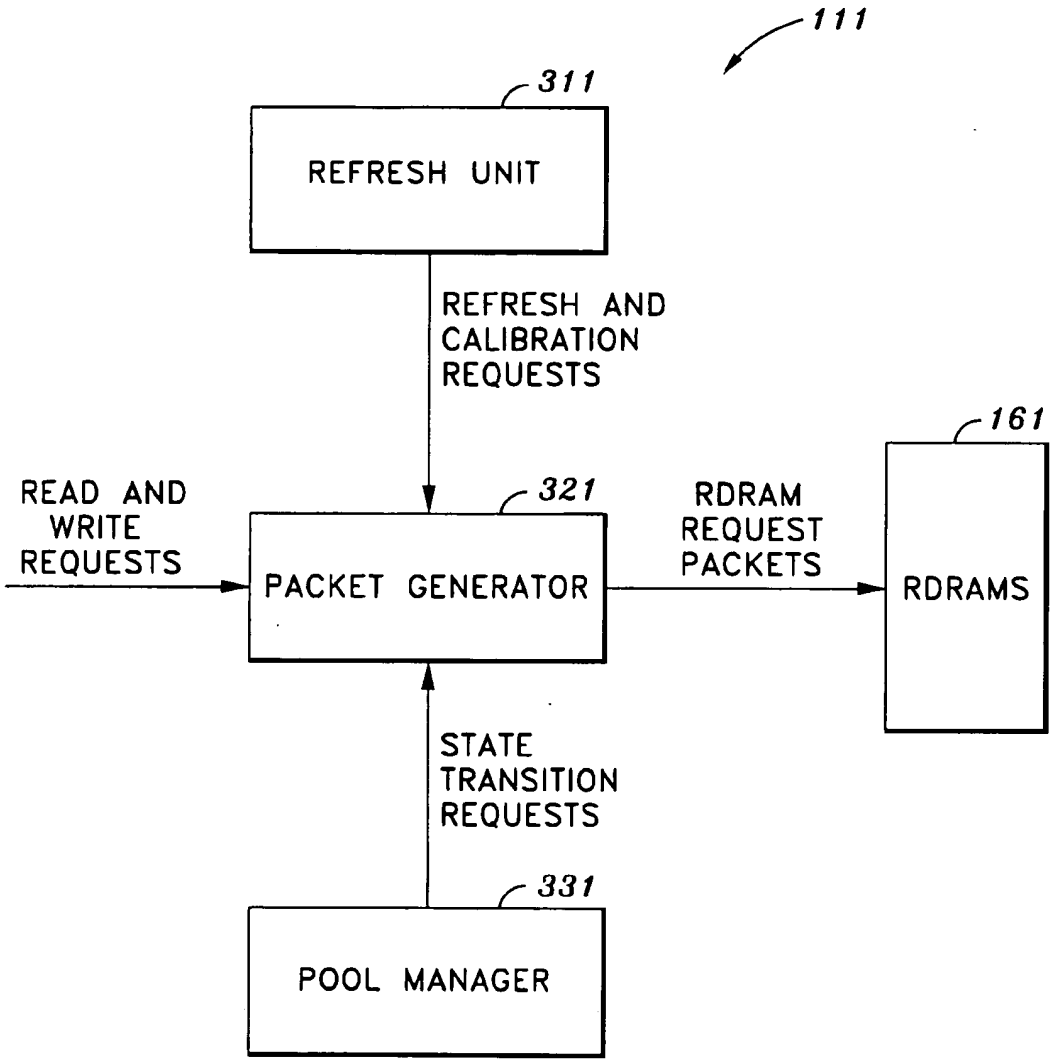

111
311
REFRESH UNIT
REFRESH AND
CALIBRATION
REQUESTS
161
READ AND
WRITE
REQUESTS
321
RDRAM
REQUEST
PACKETS
PACKET GENERATOR
RDRAMS
STATE
TRANSITION
REQUESTS
331
POOL MANAGER

FIG. 3

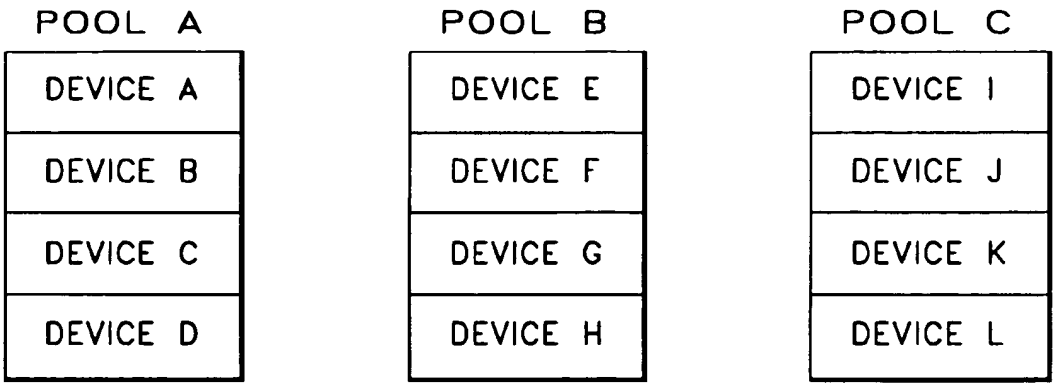


FIG. 4A

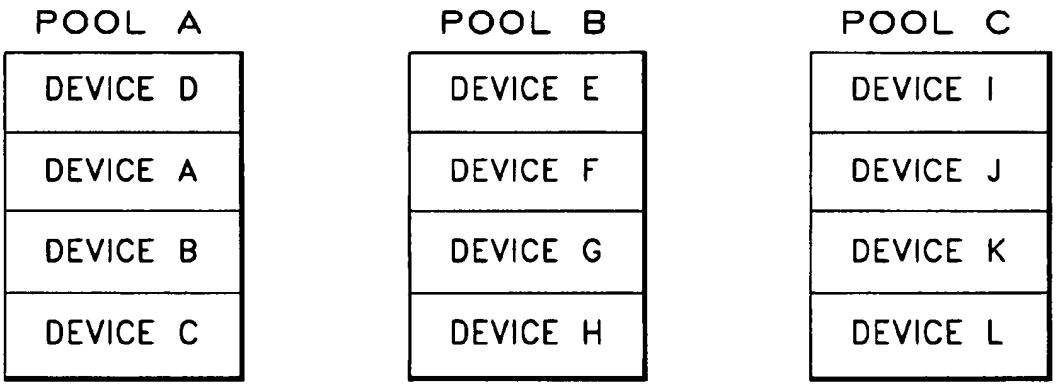


FIG. 4B

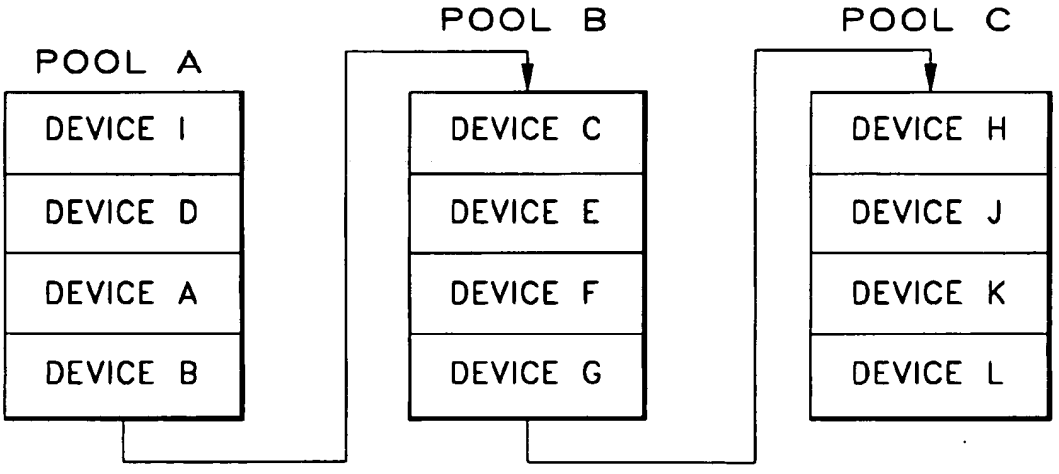


FIG. 4C

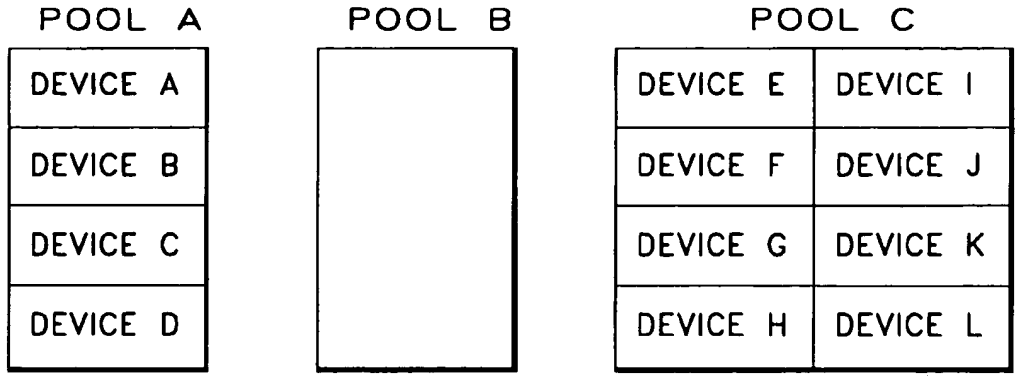


FIG. 5A

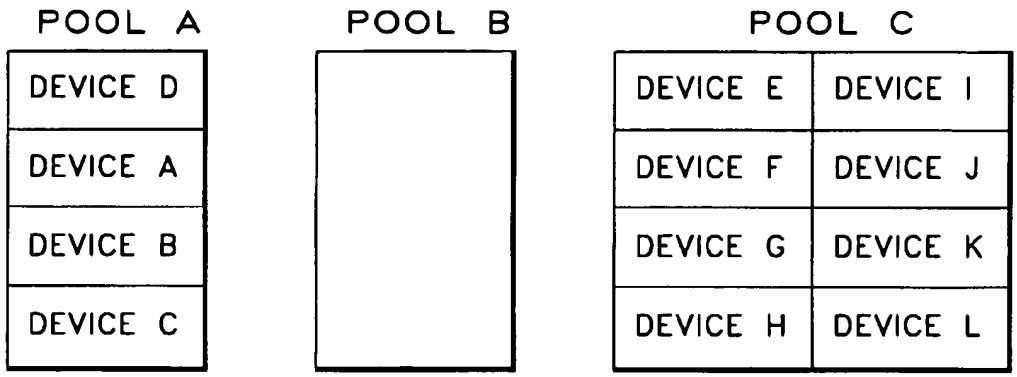


FIG. 5B

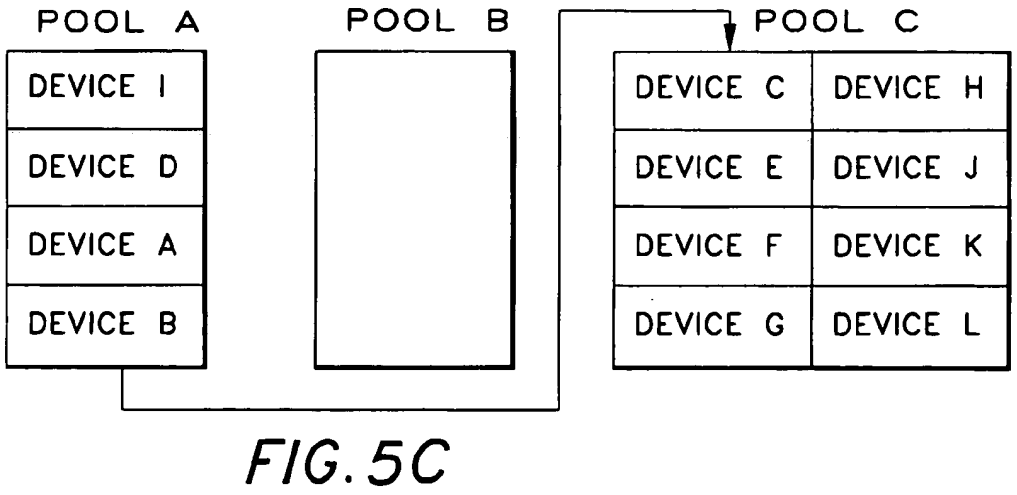


FIG. 5C

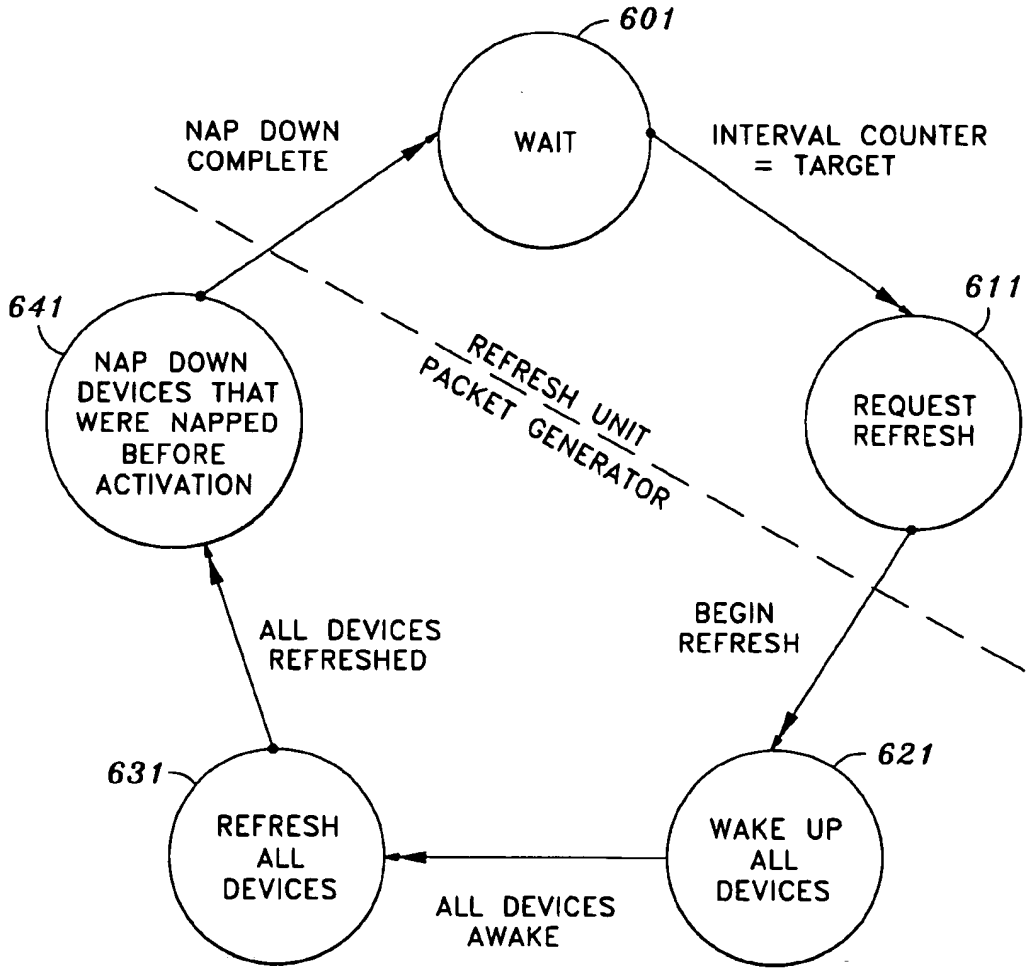

601
WAIT
NAP DOWN COMPLETE
INTERVAL COUNTER = TARGET
641
NAP DOWN DEVICES THAT WERE NAPPED BEFORE ACTIVATION
611
REQUEST REFRESH
REFRESH UNIT PACKET GENERATOR
ALL DEVICES REFRESHED
BEGIN REFRESH
631
REFRESH ALL DEVICES
621
WAKE UP ALL DEVICES
ALL DEVICES AWAKE

FIG. 6

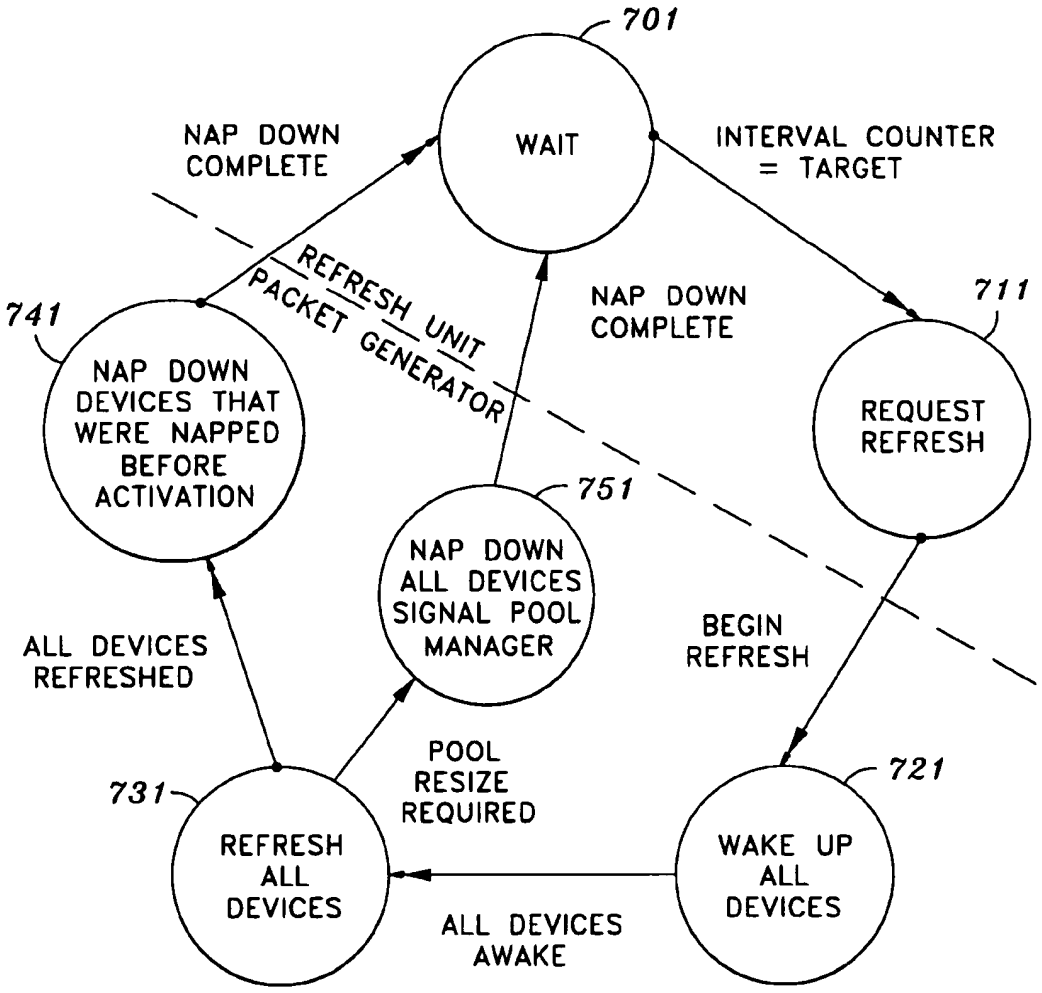

701
WAIT
NAP DOWN COMPLETE
INTERVAL COUNTER = TARGET
REFRESH UNIT PACKET GENERATOR
741
NAP DOWN DEVICES THAT WERE NAPPED BEFORE ACTIVATION
NAP DOWN COMPLETE
711
REQUEST REFRESH
751
NAP DOWN ALL DEVICES SIGNAL POOL MANAGER
ALL DEVICES REFRESHED
BEGIN REFRESH
POOL RESIZE REQUIRED
731
REFRESH ALL DEVICES
721
WAKE UP ALL DEVICES
ALL DEVICES AWAKE

FIG. 7

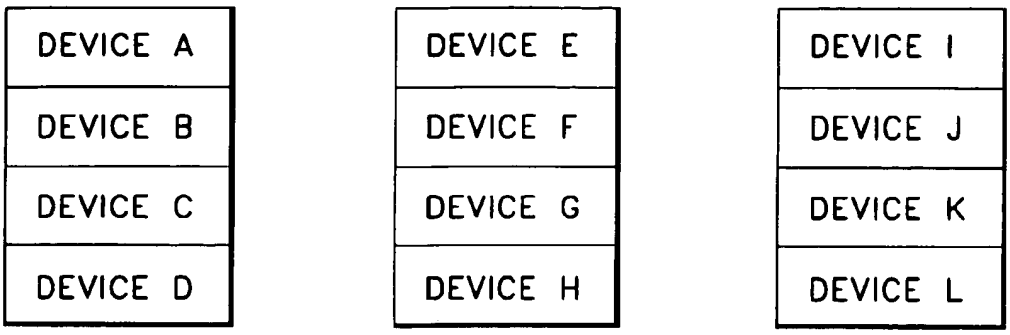

DEVICE A
DEVICE B
DEVICE C
DEVICE D
DEVICE E
DEVICE F
DEVICE G
DEVICE H
DEVICE I
DEVICE J
DEVICE K
DEVICE L

FIG. 8A

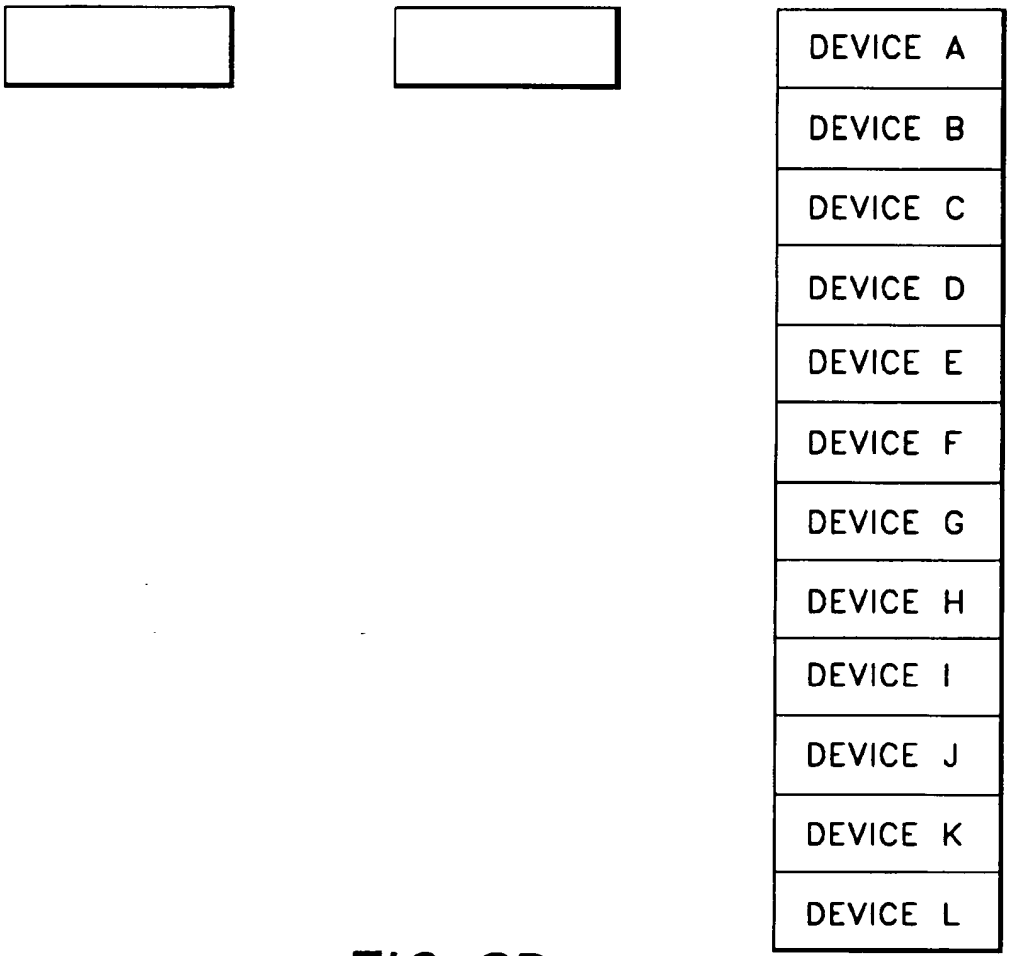

DEVICE A
DEVICE B
DEVICE C
DEVICE D
DEVICE E
DEVICE F
DEVICE G
DEVICE H
DEVICE I
DEVICE J
DEVICE K
DEVICE L

FIG. 8B

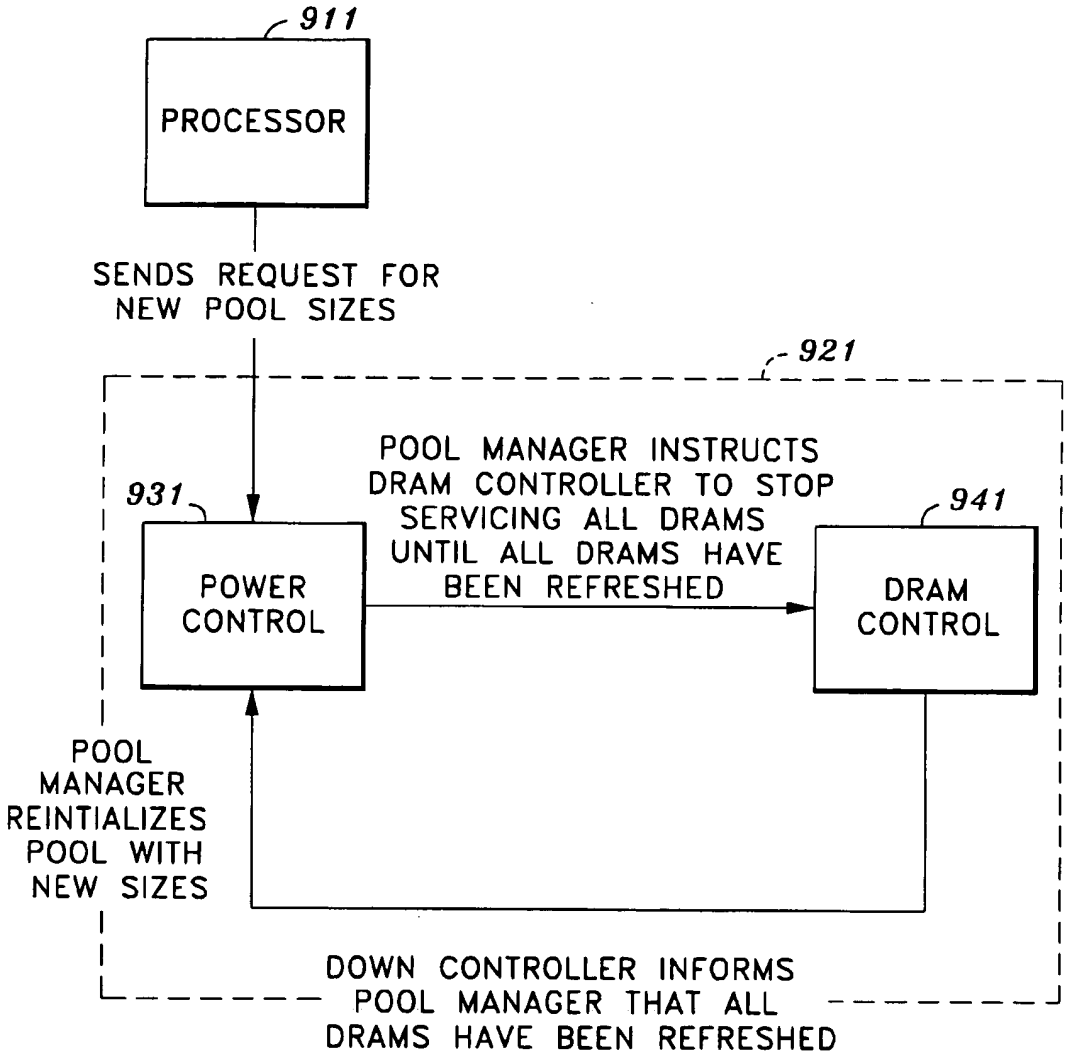

911
PROCESSOR
SENDS REQUEST FOR NEW POOL SIZES
921
931
POWER CONTROL
POOL MANAGER INSTRUCTS DRAM CONTROLLER TO STOP SERVICING ALL DRAMS UNTIL ALL DRAMS HAVE BEEN REFRESHED
941
DRAM CONTROL
POOL MANAGER REINTIALIZES POOL WITH NEW SIZES
DOWN CONTROLLER INFORMS POOL MANAGER THAT ALL DRAMS HAVE BEEN REFRESHED

FIG. 9

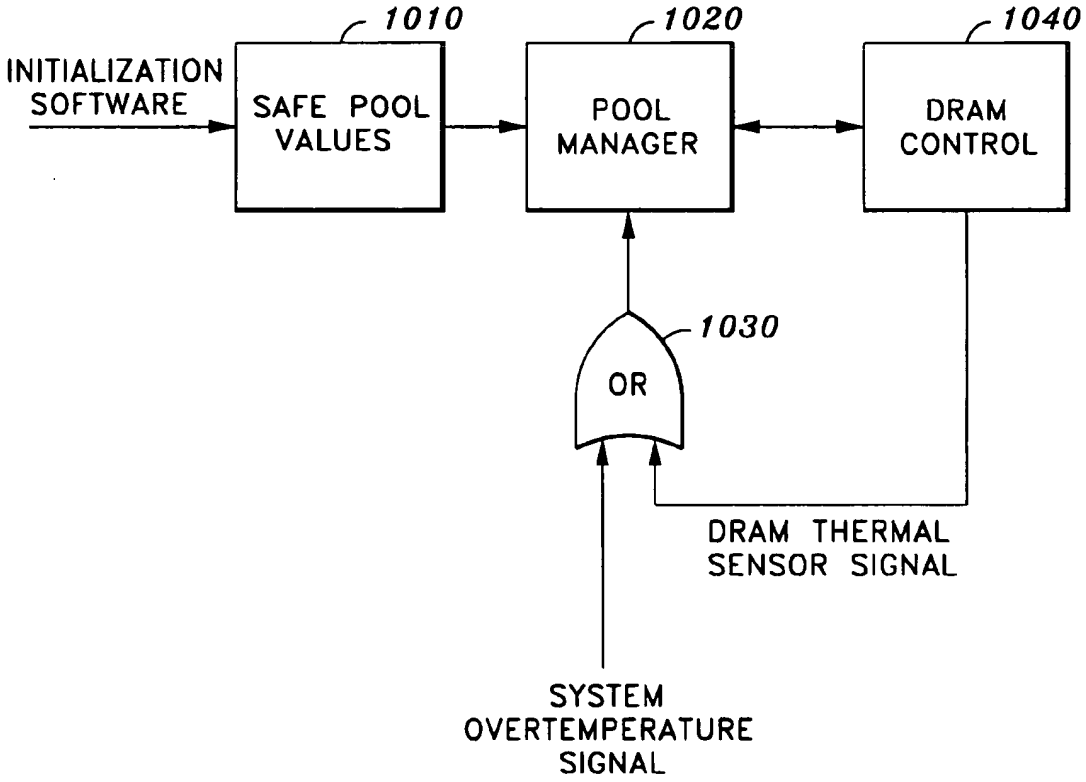

1010
1020
1040
INITIALIZATION SOFTWARE
SAFE POOL VALUES
POOL MANAGER
DRAM CONTROL
1030
OR
DRAM THERMAL SENSOR SIGNAL
SYSTEM OVERTEMPERATURE SIGNAL

FIG. 10

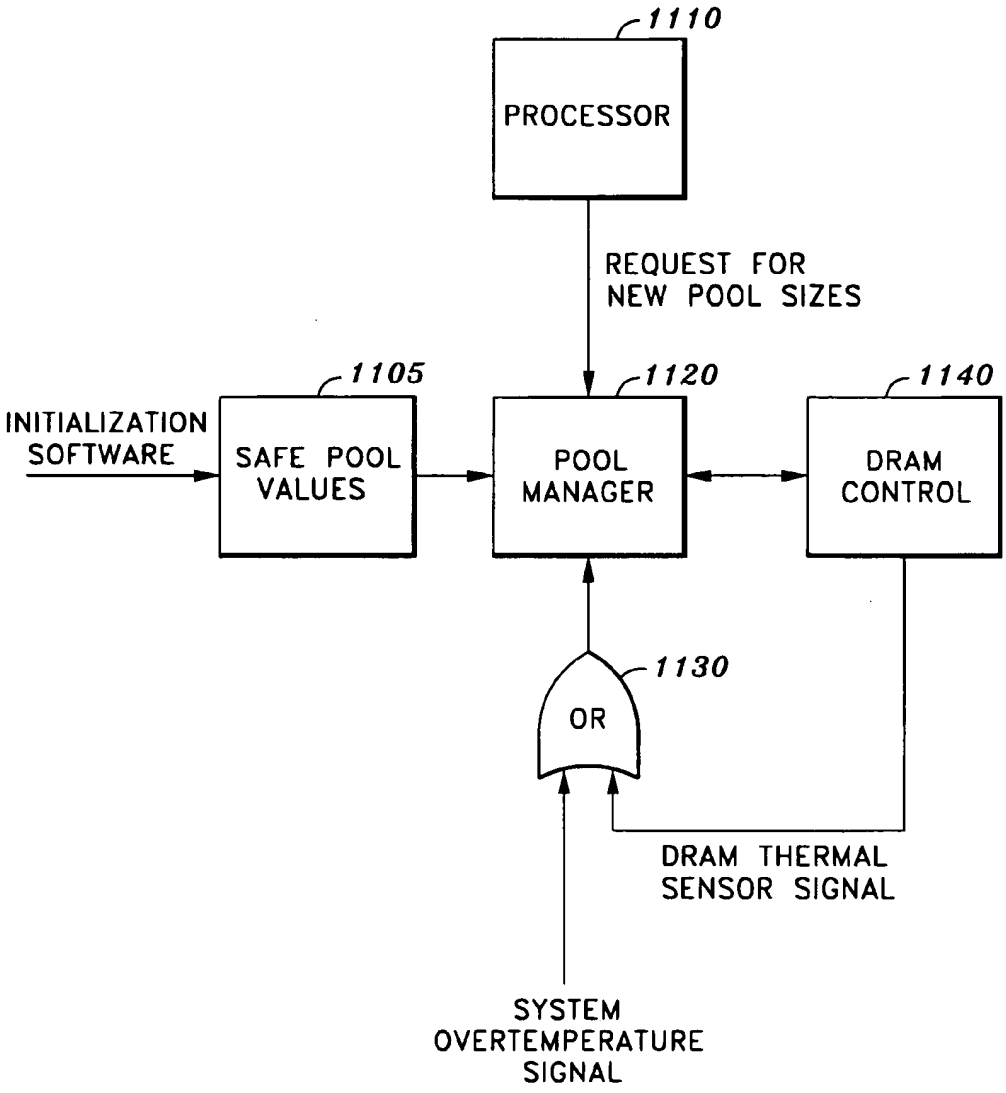

1110
PROCESSOR
REQUEST FOR
NEW POOL SIZES
1105
1120
1140
INITIALIZATION
SOFTWARE
SAFE POOL
VALUES
POOL
MANAGER
DRAM
CONTROL
1130
OR
DRAM THERMAL
SENSOR SIGNAL
SYSTEM
OVERTEMPERATURE
SIGNAL

FIG. 11

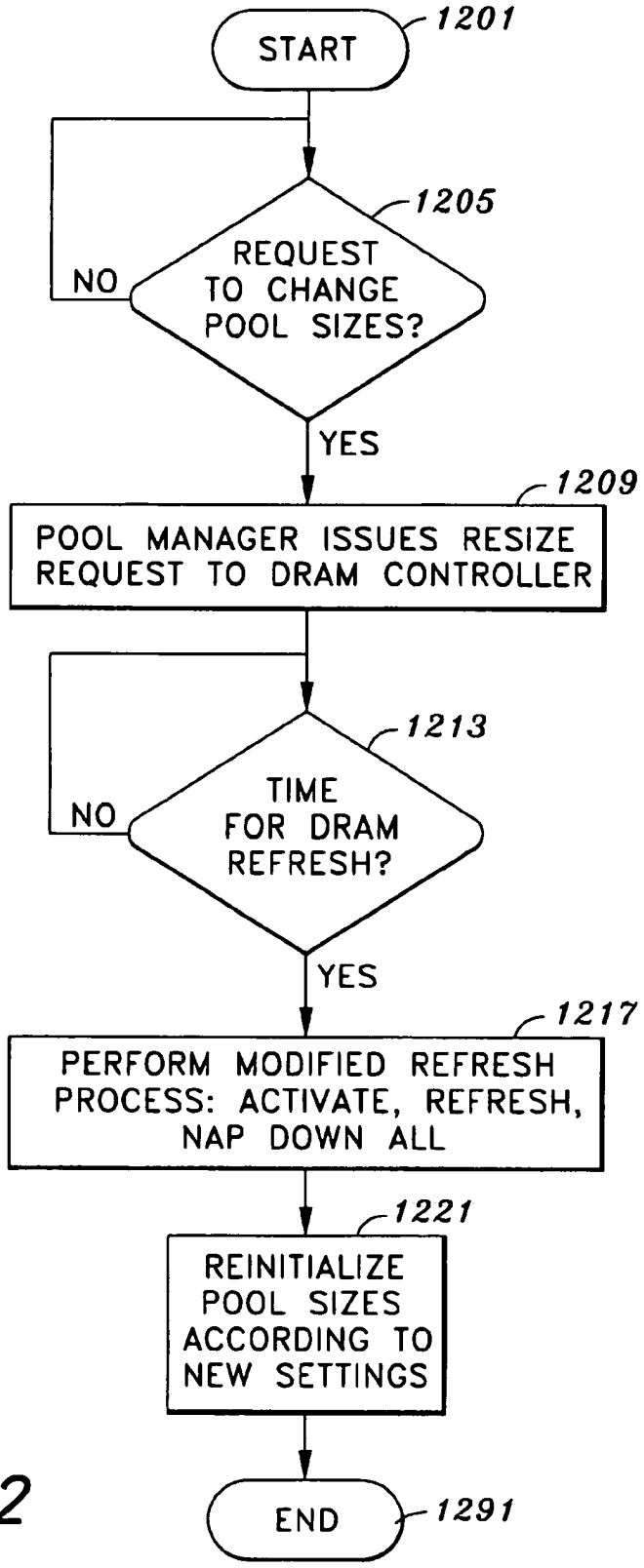

START
1201
REQUEST TO CHANGE POOL SIZES?
1205
NO
YES
POOL MANAGER ISSUES RESIZE REQUEST TO DRAM CONTROLLER
1209
TIME FOR DRAM REFRESH?
1213
NO
YES
PERFORM MODIFIED REFRESH PROCESS: ACTIVATE, REFRESH, NAP DOWN ALL
1217
REINITIALIZE POOL SIZES ACCORDING TO NEW SETTINGS
1221
END
1291

FIG. 12

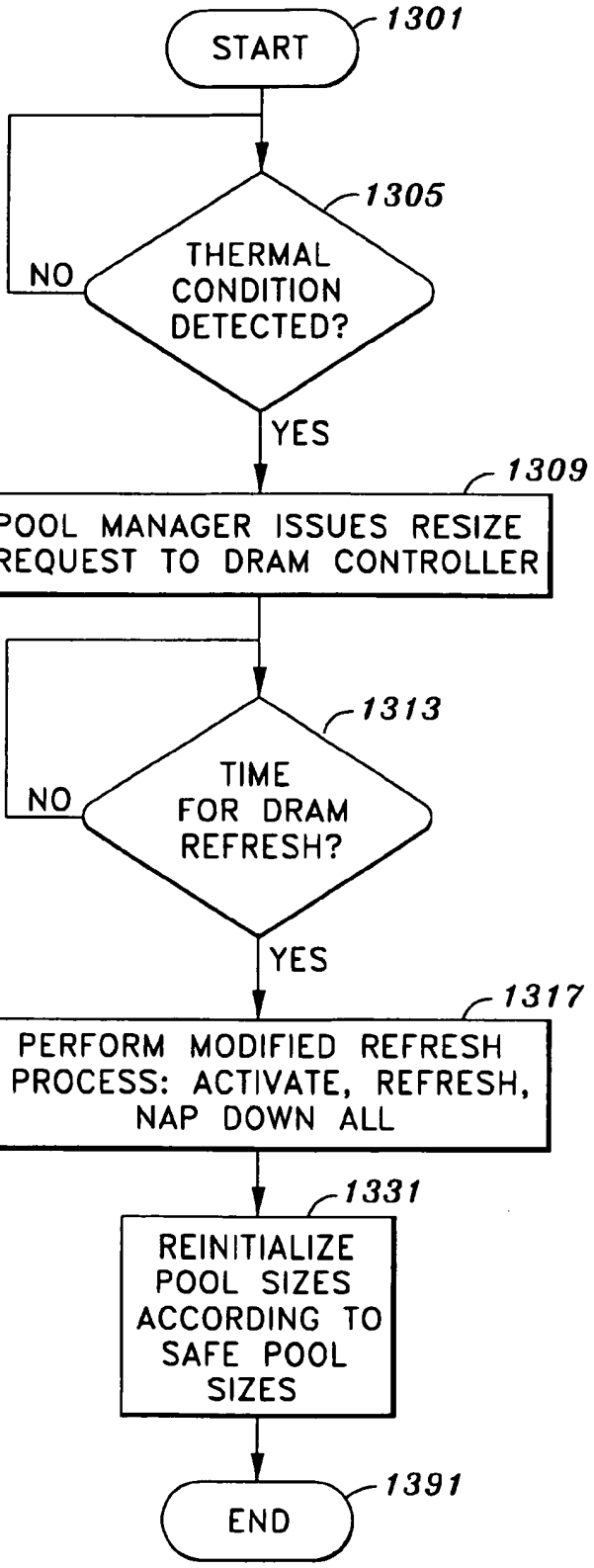

START
1301
1305
THERMAL CONDITION DETECTED?
NO
YES
1309
POOL MANAGER ISSUES RESIZE REQUEST TO DRAM CONTROLLER
1313
TIME FOR DRAM REFRESH?
NO
YES
1317
PERFORM MODIFIED REFRESH PROCESS: ACTIVATE, REFRESH, NAP DOWN ALL
1331
REINITIALIZE POOL SIZES ACCORDING TO SAFE POOL SIZES
END
1391

FIG. 13

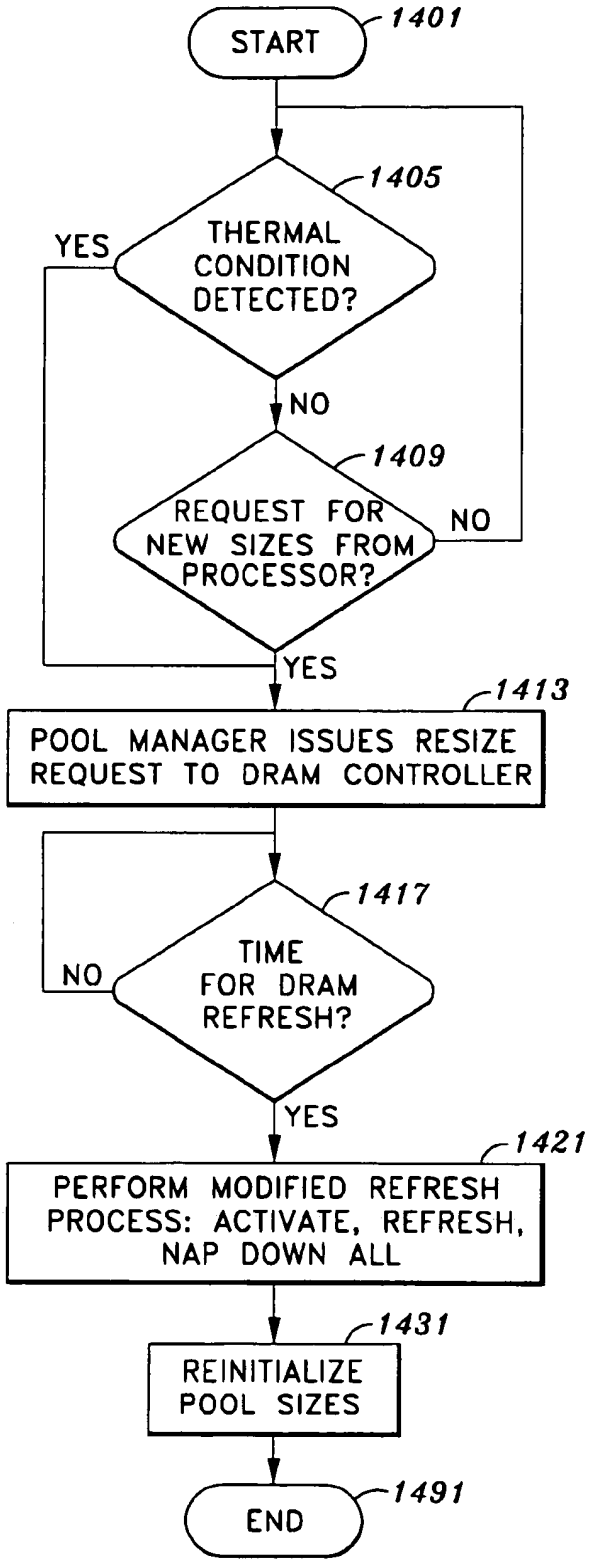

START
1401
1405
THERMAL CONDITION DETECTED?
YES
NO
1409
REQUEST FOR NEW SIZES FROM PROCESSOR?
NO
YES
1413
POOL MANAGER ISSUES RESIZE REQUEST TO DRAM CONTROLLER
1417
TIME FOR DRAM REFRESH?
NO
YES
1421
PERFORM MODIFIED REFRESH PROCESS: ACTIVATE, REFRESH, NAP DOWN ALL
1431
REINITIALIZE POOL SIZES
END
1491

FIG. 14

METHOD AND APPARATUS FOR DYNAMICALLY CHANGING THE SIZES OF POOLS THAT CONTROL THE POWER CONSUMPTION LEVELS OF MEMORY DEVICES

FIELD OF THE INVENTION

The present invention relates to the power and thermal management of computer systems and devices. More specifically, the present invention relates to an apparatus, method, and system for dynamically controlling the power consumption levels of memory devices in a memory system.

BACKGROUND OF THE INVENTION

As computer devices and systems continue to advance and become more complex, effective and efficient power and thermal management of computer devices and systems have become more and more critical in system design and implementation. Since computer devices and systems can only operate properly and safely within certain electrical power and temperature ranges, it is important to ensure that there is sufficient power supply to operate various devices when they are needed. In addition, it is also important to ensure that thermal conditions do not exceed some threshold levels that are considered safe for the operations of these various devices. In general, computer devices such as memory devices are designed to have different operating modes or power states that correspond to different levels of performance and power consumption. The different operating modes or power states may include, for example, active mode, standby mode, nap mode, etc. Generally, devices operate faster in active mode than they do in other modes. However, devices also consume more power and generate more heat in active mode than they do in other modes. Keeping all devices in the system in active mode reduces operational latency and therefore improves system overall performance. However, keeping all devices in active mode consumes more power and generates more heat dissipation. In addition, even if the system power supply source is sufficient to power all devices in the system, some of these devices may be idle anyway and therefore it would be a waste of resources to keep them in active mode all the time. System performance requirements and system power usage requirements need to be balanced. To maintain a balance between system performance and system power usage and heat dissipation, it is necessary to keep some number of devices in an inactive mode to reduce power usage and heat dissipation. Depending on the applications and the operational environment, the number of devices to be kept in inactive mode may vary.

The system constraints and tradeoffs described above with respect to computer devices in general apply equally to memory devices in memory system. In their active or most power-hungry mode, memory devices such as dynamic random access memory (DRAM) devices operate faster than they do when they are in inactive mode (e.g., standby or nap mode). However, DRAM devices in their active mode also consume far more power than they do when they are in inactive mode. As a result, to maintain a balance between performance and power consumption (and heat dissipation), some fixed number of DRAM devices may need to be kept in an inactive mode to conserve power and reduce heat dissipation. The number of devices in active mode and the number of devices in inactive mode can be specified by the Basic Input/Output Program (BIOS) at system start up (boot) or system reset. Management of which devices are in active mode and which devices are in inactive mode can be accomplished through a definition of pools of devices that are used to keep track of the operational mode or power state (e.g., active or inactive) of the individual devices. A pool of devices in this context refers to a mapping or list of devices that are in a specific operational mode or power state. For example, one pool may be maintained to keep track of the devices that are in active mode and another pool may be maintained to keep track of the devices that are in inactive mode. Under such a power management scheme, the devices represented in one of the pools are assumed to be operating in a certain operational mode or power state and therefore consuming a certain amount of power. For example, devices that are represented in the active pool are assumed to be operating in active mode. The number of devices in each pool may be examined to determine the amount of power being used by the entire memory system. The different pools utilized to keep track of the operational mode or power state of the various memory devices are also referred to as the power-control or power-saving pools hereinafter.

Conventionally, the number of devices in each pool (also referred to as the size of the pool or pool size hereinafter) is configured or specified by the BIOS at start up or reset and left unchanged during system operations because of the complexity of accounting for the power consumption states of all devices during any proposed transition. For example, a system operator or system user may specify through BIOS setup that the number of active devices is 8 and the number of inactive devices is 24. These two numbers are used to determine the maximum allowable number of devices that can be in the active and inactive pools, respectively. Such a static and inflexible pool configuration is not effective and efficient in balancing the system performance requirements with the system power and heat dissipation requirements because certain events and operating conditions may occur during the course of the system operation which could require the pool configuration to be changed for the system to continue to operate properly, safely, and efficiently. In various instances, for example, it would be useful to be able to change the pool configuration (e.g., change the size of the active pool and inactive pool, etc.) during the system operations in response to various external stimuli or changes in operational conditions since the sizes of the pools are used to maintain a proper balance between system performance and system power consumption (and heat generation). For example, the sizes of the pools may need to be changed in response to a temperature condition that exceeds the system acceptable thermal tolerances or in response to an indication that the system is operating from battery power source due to a power failure or outage. In addition, the sizes of the pools may need to be changed due to changes in the system operational characteristics such as changes in the number of system users which generally affect the usage and therefore the power consumption levels of the memory system.

As a result, there exists a need to dynamically reconfigure or change the sizes of power-control pools of memory devices during the course of the system operations.

SUMMARY OF THE INVENTION

The present invention provides a method, apparatus, and system for dynamically changing the sizes of power-control pools that are used to control the power consumption levels of memory devices. In one embodiment, a request to change the sizes of the memory power-control pools is received. In response to receiving the request to change the sizes of the memory power-control pools, the memory devices are placed in a specific operating mode or power state after

being refreshed in a periodic refresh cycle. In response to a signal indicating that all memory devices have been placed in the specific operating mode, power-control pools are resized according to pool size values corresponding to the request received.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will be more fully understood by reference to the accompanying drawings, in which:

FIG. **1** is a block diagram of one embodiment of a system implementing the teachings of the present invention;

FIG. **2** shows a block diagram of a memory controller having a memory power consumption control mechanism;

FIG. **3** illustrates a block a diagram of one embodiment of a memory control unit containing a pool manager;

FIGS. **4A–4C** illustrate an example of one embodiment of a method for managing various power-control pools that are used to keep track of and control the operational states of memory devices;

FIGS. **5A–5C** illustrate an example of one embodiment of a method for managing power-control pools that are used to keep track of and control the operational states of memory devices;

FIG. **6** shows a state diagram of one embodiment of a process for performing normal memory refresh operations;

FIG. **7** shows a state diagram of one embodiment of a process for performing refresh operations in response to a request to resize the power control pools;

FIGS. **8A–8B** show an example of a reconfiguration of the power control pools in response to a request to resize the power control pools;

FIG. **9** is a block diagram of one embodiment of an apparatus for dynamically changing the sizes of memory power-control pools;

FIG. **10** is a block diagram of one embodiment of an apparatus for dynamically changing the sizes of memory power-control pools;

FIG. **11** shows a block diagram of one embodiment of an apparatus for dynamically changing the sizes of memory power-control pools;

FIG. **12** shows a flow diagram of one embodiment of a method for dynamically changing the sizes of memory power-control pools in response to a request from a processor;

FIG. **13** illustrates a flow diagram of one embodiment of a method for dynamically changing the sizes of memory power-control pools in response to hardware-detected system events; and

FIG. **14** illustrates a flow diagram of one embodiment of a method for dynamically changing the sizes of memory power-control pools in response to a processor's request or hardware-detected system events.

DETAILED DESCRIPTION

In the following detailed description numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be understood and practiced without these specific details.

In the discussion below, the teachings of the present invention are utilized to implement a method and apparatus for dynamically changing the sizes of memory power-control pools that are used to keep track of and control the operational states of various memory devices. In one embodiment, the refresh process normally used to refresh the various memory devices is modified to make the various memory devices enter a specific operational state, e.g., the nap state, after being refreshed in response to a request to change the sizes of the memory power-control control pools. After the various memory devices have entered the specific operational state (e.g., the nap state), the sizes of the memory power-control pools can be changed to new values according to the request. In one embodiment, the request to change the sizes of the memory power-control pools can be initiated by a processor or other units that have a need for changing the sizes of the memory power-control pools. In one embodiment, the request to change the sizes of the memory power-control pools can also be initiated, without system software intervention, in response to detecting a specified system event, for example a signal from a system thermal control unit or a memory thermal control unit indicating that temperature has exceeded a threshold level. The teachings of the present invention are applicable to any scheme, method and system for power management of memory devices. However, the present invention is not limited to the power and thermal management of memory devices and can be applied to the power and thermal management of other devices.

FIG. **1** shows a block diagram of one embodiment of a system configuration in which the teachings of the present invention are implemented. The system configuration **100** includes a plurality of central processing units (CPUs) **101***a–d*, a memory control hub (also referred to as memory control unit) **111**, a P64 control unit **121**, an Input/Output (IO) control unit **131**, a graphics controller **141** coupled to a graphics subsystem **151**, and a plurality of memory devices **161**. For the purposes of the present specification, the term "processor" or "CPU" refers to any machine that is capable of executing a sequence of instructions and shall be taken to include, but not be limited to, general-purpose microprocessors, special purpose microprocessors, multimedia controllers and microcontrollers, etc. In one embodiment, the CPUs **101***a–101d* are general-purpose microprocessors that are capable of executing an Intel Architecture instruction set. The CPUs **101***a–101d*, the P64 control unit **121**, the IO control unit **131**, and the AGP graphics control unit **141** access the system memory devices **161** via the memory control unit **111**. The memory control unit **111**, in one embodiment, is responsible for servicing all memory transactions that target the system memory devices **161**. The memory control unit **111** can be a stand-alone unit, an integrated part of a chipset, or a part of some larger unit that control the interfaces between various system components and the system memory devices **161**. The P64 control unit **121** provides the interface control between a plurality of PCI-64 slots **125** and the memory control unit **111**. The IO control unit **131** provides the interface control between the memory unit **111** and various IO devices and ports including the PCI slots and PCI agents **133**, a plurality of USB ports **135**, a plurality of IDE ports **137**, and other IO devices **139**. The AGP graphics control unit **141** provides the interface control between the graphics subsystem **151** and the memory control unit **111**. The structure and functions of the memory control unit **111** are described in greater details below.

For the purposes of the present specification, the memory devices **161** are assumed to be dynamic random access memory (DRAM) devices. It is well known that DRAM is

a type of RAM that uses volatile storage cells which are periodically refreshed in order to hold data. The rate of refresh or frequency of refresh varies depending upon the type DRAM used, the amount of memory installed, the configuration of the system memory, etc. In the discussion that follows, it is also assumed that the memory devices used are RAMBUS® DRAMs (also referred to as RDRAMS) that are designed by Rambus Inc. of Mountain View, Calif. Everything discussed herein, however, is equally applicable to other types of DRAMs including conventional DRAMs, fast page mode (FPM) DRAMs, extended data out (EDO) DRAMs, burst extended data out (BEDO) DRAMs, synchronous DRAMs (SDRAMs), double data rate SDRAMs (DDR SDRAMs), synchronous-link DRAM (SLDRAMs), etc.

FIG. **2** shows a block diagram of one embodiment of the memory control unit **111** described in FIG. **1**. In this embodiment, the memory control unit **111** contains three major blocks, the host group (HG) **211**, the IO group (IOG) **221**, and the data group (DG) **231**. In one embodiment, the host group **211** functions as a host interface for the memory control **111**. Some of the functions performed by the host group **211** include receiving transaction requests from the CPUs **101***a*–**101***d*, generating appropriate commands to both the IO group **221** and the data group **231**, receiving responses from the IO group **221** and the data group **231**, and transmitting the responses received to the host (CPUs **101***a*–**101***d*). In addition, the host group **211** is also responsible for generating snoop requests to the data group **231**, receiving snoop responses from the data group **231**, and transmitting snoop responses to the host. The IO group **221**, in one embodiment, functions as an IO interface for the memory control unit **111**. More specifically, the IO group **221** handles the interface functions between the data group **231** and the P64 control unit **121**, the **10** control unit **131**, and the graphics control unit **141**. In one embodiment, the data group (also referred to as data cluster) **231** is responsible for dispatching and completing all memory transactions that target the system RDRAMs. In one embodiment, the data group **231** contains two logical subcomponents: a data unit (Dunit) that performs the intricate mechanics of sending transactions to the RDRAM devices via the RAMBUS channel controller (RAC) and the buffer unit (Bunit) that is responsible for sequencing, buffering, and delivering data that are pulled from or pushed to the RDRAM devices across the memory bus (also referred to as the RAMbus). The Dunit accepts memory read, write, and refresh requests from the Bunit. These requests are decoded to determine the status of the memory pages to which they are targeted. The Dunit then generates the appropriate commands or instructions (also called the packets) necessary to carry out the memory access requests and queues up the packets for transmission across the memory bus. In addition, the Dunit also synchronizes data transfers that cross a clock boundary between the core frequency and the base frequency of the memory bus. The Bunit, in one embodiment, receives requests for memory data from the host group **211** and the IO group **221** and generates the appropriate memory access requests to the Dunit as described above.

FIG. **3** shows a block diagram of one embodiment of the memory control unit **111** that contains a refresh unit **311**, a packet generator **321**, and a pool manager **331**. The functions of these units and the interactions between them are described in more details below. As mentioned above, the memory control unit (MCU) **111** is responsible for handling memory transactions received from various sources in a timely fashion. The memory transactions received from various sources within the system **100** include memory data read and write requests. In one embodiment, the MCU **111** translates the read and write requests received from the various sources into commands that are understood by the RDRAM devices that are coupled to the MCU **111** via the memory bus. The commands understood by the RDRAM devices (i.e., the RDRAM native requests) are also called RDRAM request packets or simply packets herein. The packet generator unit **321** shown in FIG. **3** is the unit within the MCU **111** that is responsible for creating and sending packets to the RDRAM devices.

In one embodiment, the MCU **311** is also responsible for RDRAM maintenance operations such as refresh and calibration. As explained above, RDRAMs, like any other DRAM technology, uses volatile storage cells which must be refreshed periodically in order to hold data. The MCU **311** perform these maintenance operations at regular intervals by sending packets to the RDRAMs to instruct them to refresh their data or to calibrate their electrical characteristics. In one embodiment, the MCU **111** uses the refresh unit **311** shown in FIG. **3** to perform the RDRAM maintenance operations. In one embodiment, the refresh unit **311** maintains a counter used to keep track of the intervals of time between refresh or calibration cycles. When the refresh unit **311** has determined that a maintenance cycle needs to be performed on the RDRAMs, it places a request to the packet generator **321**, which in turn creates the appropriate RDRAM request packets that cause the RDRAM devices to perform the required maintenance functions (e.g., refresh or calibrate).

The pool manager **331** is responsible for managing the power consumption levels (also referred to herein as the operating modes or power states) of the RDRAM devices. As explained above, in order to maintain the balance between system performance and system power usage, memory devices are designed to have different operating modes that correspond to different performance levels (i.e., speeds). In the present embodiment, the RDRAMs are designed to have several operating modes: active, standby, nap, and powerdown. These four different operating modes of the RDRAMs are distinguished by two factors: their power consumption levels and their performance levels. For example, a RDRAM in active mode is ready to immediately service a transaction. However, power consumption is also higher in active mode than in the other modes. The four different power consumption levels and performance levels of the RDRAMs corresponding to the four different operating modes are illustrated in table 1 below, where 4 in the power consumption column corresponds to the highest level of power consumption by the RDRAM and 1 in the performance level column corresponds to the highest level of performance.

TABLE 1

RDRAM Operating Mode (Power State)	RDRAM Power Consumption Level (4 = highest, 1 = lowest)	RDRAM Performance Level (1 = fastest, 4 = slowest)
Active	4	1
Standby	3	2
Nap	2	3
Power-Down	1	4

As illustrated in table 1, RDRAMs operate faster in active mode than in the other three modes. However, RDRAMs also consume much more power in active mode than in the other three modes. Power consumption and also heat pro-

duction of memory devices (e.g., RDRAMs in the present discussion) can be reduced by placing one or more RDRAMs in a lower power mode (e.g., standby, nap, or power-down mode). As explained above, power and thermal management in modem and often complex computer systems have become more and more critical in system design and implementation. To achieve some acceptable balance between system performance and system power consumption (which also corresponds to heat dissipation), systems are typically configured so that only a fixed number of memory devices (e.g., RDRAMs) is allowed to operate in active mode. As explained above, depending on the applications and system operational environments, the number of memory devices to be kept in active mode varies. For example, in a system configuration using 12 RDRAM memory devices, certain system constraints may dictate that only a maximum of 4 RDRAM devices can be allowed to be active at any given time. As described above, the maximum number of devices in active mode, in standby mode, or in nap mode, etc., can be specified by the system user through system BIOS at system start up or reset. Management of which devices are in which operating mode (e.g., active, standby, nap, etc.) can be accomplished using a definition of pools of devices (also referred to as power-control pools) that are used to keep track of and control the operating mode or power state of the individual memory devices. A pool in the present discussion refers to a mapping or list of memory devices that are in a specific operating mode or power state.

Continuing with the present discussion, as noted above, RDRAMs can consume a substantial amount of power and generate a substantial amount of heat when operating in active mode. As a result, it would be beneficial to operate as many RDRAMs as are practical in a low power state. In one embodiment, the MCU **111** accomplishes this throttling by placing a number of memory devices (e.g., the RDRAMs) into nap mode in which the memory devices consume much less power and therefore generate much less heat than they do in active mode or standby mode. RDRAMs in nap mode can retain their data but are unable to provide their data to the MCU **111** until they are moved into either active or standby mode. As explained above, to maintain a balance between system performance and power consumption, only some fixed number of memory devices should be put into nap mode at any given time. Consequently, only some fixed number of memory devices are to be kept in active or standby mode at any given time. Computer programs are unaware of the operating mode or power state of any given memory device. Therefore, the operating mode of a given memory device may need to be changed by the MCU before that particular memory device can service a memory transaction.

Referring again to FIG. **3**, the pool manager **331** within the MCU **111** is the unit responsible for maintaining the balance between the power consumption of the RDRAM devices and their corresponding performance levels. More specifically, the pool manager **331** keeps track of the operating mode of each individual memory device and takes appropriate actions to move the memory devices from one operating mode to another based upon various factors including the maximum number of devices allowed in each operating mode, which device is required to service a particular memory transaction, etc. In one embodiment, in order to change the operating mode of a particular memory device, the pool manager **331** requests the packet generator to send the appropriate packets (i.e., commands) to the memory device that instruct the memory device to perform the required function (e.g., changing from active to standby mode or changing from standby to nap mode, etc.).

As described above, in one embodiment, the pool manager **331** maintains information about the operating mode of each individual devices (i.e., which devices are in active, standby, or nap modes) through the use of a plurality of pools where each pool refers to a mapping or list of devices that are in a specific operating mode or power state). In one embodiment, the pool manager **331** uses three pools to keep track of the operating modes of the memory devices. One of the pools, called the active pool or pool A, is used to keep track of which devices operating in active mode. The other pool, called the standby pool or pool B, is used to keep track of which devices operating in standby mode. The remaining pool, called the nap pool or pool C, is used to keep track of which devices operating in nap mode. Each of the three pools therefore contains references to the devices that are in a specific operating mode or power state. In one embodiment, the information in the active pool and the standby pool are stored in a set of registers while the nap pool is represented by the subtractive subset of the memory devices that are not found in either the active pool or the standby pool. While the teachings of the present invention are discussed herein using pools as an example of a mechanism for maintaining information about the operating modes of memory devices, it would be obvious to one skilled in the art that the present invention equally applies to other methods, mechanisms or protocols for maintaining and controlling operating modes of the memory devices.

In one embodiment, the MCU **111** can have two modes of operations with respect to the operating mode configuration of the RDRAM devices. In the first mode, all devices are assumed to be in either in active or standby mode. In this configuration, all active devices are represented by tokens in pool A (the active pool), pool B is unused, and pool C subtractively contains all devices that are not represented in pool A. As a result, all devices represented in pool C are assumed to be in standby mode. In the second mode of operation, the memory devices may be in active, standby, or nap modes. All three pools A, B, and C are utilized in this configuration. Pool A represents all active devices, pool B represents devices that are in standby mode, and pool C subtractively is used to represent all devices that are neither in pool A nor pool B and are therefore assumed to be in nap mode.

In one embodiment, the pool manager **331** employs true least-recentlyused (LRU) algorithm to maintain the list of devices represented in pools A and B. FIGS. **4A–4C** illustrate an example of the configuration and maintenance of the three pools A, B, and C when the MCU **111** operates in the second operation mode (i.e., the memory devices may be in active, standby, or nap modes). In this example, both pool A and pool B are assumed to be set to the sizes of **4** and may therefore each represent up to four memory devices. It is also assumed in this example that there are **12** memory devices in the system, labeled from A to L.

FIG. **4A** shows a configuration of the three pools A, B, and C at some given point in the course of system operations. At this stage, as shown in FIG. **4A**, devices A–D are represented in pool A and therefore assumed to be in active mode. Devices E–H are represented in pool B and therefore assumed to operate in standby mode. Devices I–L are represented in pool C and therefore assumed to be in nap mode. In the present example, the device listed at the top of the list (e.g., device A in FIG. **4A**) is considered the most recently used while the device listed at the bottom of the list (e.g., device D in FIG. **4A**) is considered the least recently used. Using the pool representation shown in FIG. **4A**, the three pools A, B, and C would be transformed to those

represented in FIG. 4B after a read or write to a location in device D. The "D" token representing device D would move to the most recently used position (the top of the list) in pool A, while pools B and C would be unaffected since the change with respect to device D did not affect the number of devices allowed in each pool. Assuming that device I were accessed next, then the three pools A, B, and C would be transformed to those shown in FIG. 4C. In this case, device I was moved to the most recently used position in pool A. Because device I was changed from nap mode to active mode, device C which was the least recently used device in pool A was moved to the most recently used position in pool B in order to maintain the maximum allowable number of active devices in pool A. Similarly, since device C was changed from active mode to standby mode, the least recently used device in pool B (i.e., device H) was kicked out of pool B and subtractively moved to pool C in order to maintain the maximum allowable number of devices in pool B. FIG. 4C therefore represents the pool representation of the three pools A, B, and C after the corresponding devices have changed to their appropriate operating modes.

FIGS. 5A–5C illustrate an example of the configuration and maintenance of the three pools A, B, and C when the MCU **111** operates in the first mode (i.e., the memory devices are assumed to operate either in active or standby mode). In this example, pool A is assumed to have the maximum size of four which means that up to four devices are allowed to be active at any given time. Pool B is not used. It is also assumed in this example that there are **12** memory devices in the system, labeled from A to L.

FIG. **5A** shows a configuration of the three pools A, B, and C at some given point in the course of system operations. At this stage, as shown in FIG. **5A**, devices A–D are represented in pool A and therefore assumed to be in active mode. Devices E–L are subtractively represented in pool C (i.e., since these devices are not represented in pool A, they are assumed to be in standby mode). Again, in the present example, the device listed at the top of pool A (e.g., device A in FIG. **5A**) is considered the most recently used while the device listed at the bottom of pool A (e.g., device D in FIG. **5A**) is considered the least recently used. Using the pool representation shown in FIG. **5A**, the three pools A, B, and C would be transformed to those represented in FIG. **5B** after a read or write to a location in device D. The "D" token representing device D would move to the most recently used position (the top of the list) in pool A, while pools B and C would be unaffected since the change with respect to device D did not affect the number of devices allowed in pool A. Assuming that device I were accessed next, then the three pools A, B, and C would be transformed to those shown in FIG. **5C**. In this case, device I was moved to the most recently used position in pool A. Because device I was changed from standby mode to active mode, device C which was the least recently used device in pool A was kicked out of pool A and subtractively moved to pool C in order to maintain the maximum allowable number of active devices in pool A. FIG. **5C** therefore represents the pool representation of the three pools A, B, and C after the corresponding devices have changed to their appropriate operating modes.

As explained above, to maintain a balance between the system performance, power consumption and thermal safety, it is necessary to keep some number of memory devices in active mode and the rest of devices in lower power states (e.g., standby or nap mode). More specifically, since the number of memory devices operating in each of the power states affects the performance, power consumption and heat production levels of the system, it is useful to keep the sizes of the active, standby, and nap pools within some initial threshold limits to maintain some balance between system performance, power consumption and heat production. Conventionally, the maximum number of devices allowed in each pool is configured or specified using the system BIOS at start up or reset and left unchanged during the course of system operations. Such a static and inflexible pool configuration, while providing an initial system balance between performance and power consumption, is not effective and efficient in balancing the system performance requirements with the system power and heat dissipation requirements. This is because various events and operating conditions may occur during the course of the system operations which could require the initial pool configuration to be changed for the system to continue to operate properly, safely, and efficiently. In other words, the conventional method of pool configuration and control does not account for the dynamic nature of computer system operations and reliability since the changes in the system thermal environment or system usage intensity might require a change in the operating modes of the memory devices which are controlled by the pool manager **331**. In various instances, it would be useful to be able to change the sizes of the pools during system operations in response to various external stimuli or changes in operational conditions because the sizes of the pools are used to maintain a proper balance between system performance, system power consumption and heat generation. For example, system overheating could be used as a trigger to place some of the active memory devices into nap mode in order to reduce the heat generation capacity of the memory devices. In addition, the sizes of the pools may need to be changed in response to an indication that the system is operating from lower power source (e.g., battery) due to some power failure or outage. Furthermore, the sizes of the pools may need to be changed due to other changes in the system operational characteristics such as changes in the number of system users, etc.

As explained above, the pool configuration conventionally is left unchanged during system operations because of the substantial complexity created in changing the pool size during operations. This is because appropriate commands must be sent to the RDRAM devices to transition them from one state to another, in addition to the movement or update of values in the registers used to maintain the operating modes or power states of the memory devices. This is necessitated by the mandate that the states of the memory devices match the states reflected inside the MCU 111. The present invention solves this problem by exploiting a characteristic of the refresh operations performed by the refresh unit **311** which requires memory devices to be moved into known power states at regular time intervals. More specifically, in order to perform refresh operations, all memory devices are activated (i.e., transition into active mode) before refresh request packets are sent to them. The refresh unit **311**, in performing periodic refresh operations, normally asks that, upon refresh completion, the devices are restored by the packet generator **321** to their states before the activation.

FIG. **6** shows a state diagram of one embodiment of a process for performing normal refresh operations which reflects states of the memory devices that exist in both the refresh unit **311** and the packet generator **321**. The refresh unit **311** enters a wait state at block **601**. When the refresh counter that is used to keep track of the time intervals between refreshes reaches a predetermined target number, the refresh unit **311** initiates the refresh operations. At block **611**, a refresh request is initiated that begins the refresh

operations. As mentioned above, the refresh unit begins the refresh operations by sending a request to wake up or activate all memory devices to the packet generator **321** which in turn issues the appropriate packets to the memory devices. At block **621**, a request to wake up all memory devices is initiated. In response to a signal indicating that all devices have been activated, a request to refresh all memory devices is initiated at block **631**. In response to a signal indicating that all devices have been refreshed, a request to nap down devices that were in nap mode before the activation is initiated at block **641**. After the devices have been restored to their states before the activation, the refresh unit **311** re-enters the wait state at block **601** to wait for the next refresh cycle.

To allow for a dynamic configuration of the power-control pools (i.e., changing the sizes of the pools during system operations), the present invention modifies the refresh process described above so that all devices are placed in a specific mode, for example the nap mode, at the end of the refresh process instead of just napping down the devices that were in nap mode before the activation. This method allows the pool manager **331** to simply reset its pools upon the completion of a refresh operation when such a pool reconfiguration is desired during the course of system operations. In one embodiment, the modified refresh process is initiated by a request from the pool manager **331** that a pool resize or reconfiguration is required. In one embodiment, when all devices are napped down at the end of the modified refresh process, the packet generator **321** and the refresh unit signals the pool manager **331** to indicate that the pools may now be reconfigured or reinitialized. Once the pools have been reconfigured or reinitialized, the pool manager **331** can resume its normal operation since it is capable of maintaining the pools after the states of the devices are known.

FIG. **7** illustrates a state diagram of one embodiment of a modified refresh process in response to a request to reconfigure or resize the power-control pools. The refresh unit **311** enters a wait state at block **701**. When the refresh counter that is used to keep track of the time intervals between refreshes reaches a predetermined target number, the refresh unit **311** initiates the refresh operations. At block **711**, a refresh request is initiated that begins the refresh operations. As mentioned above, the refresh unit begins the refresh operations by sending a request to wake up or activate all memory devices to the packet generator **321** which in turn issues the appropriate packets to the memory devices. At block **721**, a request to wake up all memory devices is initiated. In response to a signal indicating that all devices have been activated, a request to refresh all memory devices is initiated at block **731**. In this example, it is assumed that the pool manager has initiated a request for pool resize. Because a request to resize the pools has been initiated, instead of proceeding to block **741** to restore the states of the devices that were in nap mode before the activation, the process proceeds to block **751** to nap down all memory devices at the end of the refresh and signal the pool manager when all devices have been napped down. The process then re-enters the wait state at block **701** to await the next refresh cycle. As explained above, by modifying the refresh process, dynamic pool configuration can be achieved. FIGS. **8A–8B** show an example of a configuration of the three pools A, B, and C in response to a request to resize the pools. In this example, the active pool is to be resized to one and the standby pool is to be resized to one also. FIG. **8A** shows the status of the three pools at some point during system operations. FIG. **8B** shows the status of the three pools upon the completion of a request to resize the pools. As shown in FIG. **8B**, both pools A (the active pool) and pool B (the standby pool) pool are empty and all memory devices are represented in pool C (the nap pool). The pool manager now can resume its normal operations to maintain the pools according to the new sizes of the pools.

FIG. **9** shows a block diagram of one embodiment of an apparatus for dynamically changing the sizes of the power-control pools used to maintain and control the operating modes of memory devices. In one embodiment, the power control unit **931** and the DRAM control unit **941** are contained within a memory control unit **921** that employs the teaching of the present invention described herein to dynamically resize the power-control pools. In one embodiment, the power control unit **931** contains a pool managing unit (i.e., a pool manager) that is responsible for maintaining and controlling the sizes of the power-control pools and the operating modes of the memory devices. In one embodiment, the DRAM control unit **941** functions to control the various memory transactions that target the memory devices and also to perform various maintenance operations such as refresh and calibration operations as described above. In one embodiment, the DRAM control unit **941** includes a refresh unit and a packet generator such as the ones that are described in FIG. **3**. One or more processor units (e.g., CPUs) are coupled to the power control unit **931** to send requests for pool resize. As explained above, during the course of system operations, it may be necessary or beneficial to change the sizes of the power-control pools in response to various changes in system operations and conditions, for example changes in system thermal environment or system usage levels, in order to maintain a balance between system performance, power consumption, and heat production requirements. When such a change is detected, the processor **911** initiates a request to resize the power-control pools to the power control unit **931**. In one embodiment, in response to receiving a request to resize the pools, the pool manager within the power control unit **931** will initiate a request to the DRAM control unit **941** to stop servicing memory transactions until all DRAMs have been refreshed. As described above, in this case, the pool manager will also instruct the DRAM control unit **951** that a pool resize request has been initiated and therefore all memory devices are to be napped down upon completion of the refresh. More specifically, the DRAM control unit **941** will perform the modified refresh process described above so that all memory devices will be placed in a known state (e.g., the nap mode) upon completion of the refresh. The pool manager within the power control unit **931** will then reinitialize the pools with new sizes upon receiving a signal from the DRAM control unit **941** that all memory devices have been refreshed and placed in a specific state (e.g., nap mode). After reinitializing the pools, the pool manager can now resume its normal operations to maintain and control the pools using the new pool sizes corresponding to the request received from the processor **911**. In one embodiment, the processor **911** can supply the information indicating the new sizes with the request to resize. In another embodiment, the new size information can be pre-stored and the pool manager can determine which new sizes to be used by the type of request received from the processor **911**.

Applying the teachings of the present invention which enable the dynamic configuration of the power-control pools, system software or programs can be modified or designed to monitor system thermal conditions and generate appropriate requests to the power control unit to change the sizes of the pools when necessary, as illustrated in FIG. **9**, to keep the system operation within a safe temperature region.

Unfortunately, thermally induced errors might prevent the system from operating normally and therefore may block the system software from having its chance to remedy any potentially damaging conditions. More specifically, system software may not respond fast enough or may itself become inoperable due to thermal errors. Because of this, there exists a need for dynamically changing the sizes of the power-control pools without system software interventions. To solve this problem, the teachings of the present invention can be utilized to provide for a mechanism that allows the memory controller to respond to various thermal conditions in a configurable manner that is capable of throttling down the power consumption levels of the memory devices quickly and without system software intervention. In one embodiment, the pool manager described above can be coupled to receive a signal indicating that a thermal condition exceeding some threshold level (i.e., thermal overload) has been detected. The signal indicating such a thermal overload condition can come from external hardware or thermal sensor responsible for monitoring the thermal conditions at component or system level. In addition, the signal indicating the thermal overload condition can come from other hardware that monitors the thermal conditions of the memory devices themselves. In one embodiment, either condition will cause the system hardware to respond to change the sizes of the power-control pools to some safe values in order to quickly reduce the power consumption level of the memory system. In one embodiment, safe size values of the pools can be provided by the system software and stored in a register. For example, the information indicating the sizes of the active, standby, and nap pools can be provided by the system software and stored in a register accessible by the pool manager. When a thermal overload condition is detected, the pool manager will resize the pools according to the safe size values stored in the register to reduce the power consumption level of the memory devices to a level that is considered safe. The sizes of the pools that are considered safe, of course, can vary depending on the system configuration, the power consumption levels of memory devices in different power states, the severity of the thermal overload conditions, etc. Moreover, different levels of safe size values can be provided for different system events or variations thereof.

FIG. **10** shows a block diagram of one embodiment of an apparatus for dynamically changing the sizes of the power-control pools in response to various system events without system software intervention. In one embodiment, the pool manager **1020** is responsible for controlling the sizes of the power-control pools without system software intervention. In one embodiment, one or more registers are utilized to store the sizes of the pools that are considered safe sizes to be used by the pool manager **1020** when it is necessary to resize the pools in response to some specified events indicated by the output signal of the OR gate **1030**. The safe sizes or any variations thereof, in one embodiment, can be provided or programmed by system software. In one embodiment, the inputs to the OR gate **1030** can come from two different sources. One of the two inputs to the OR gate can come from some hardware device responsible for monitoring the thermal conditions of specific components at the component level or the system at the system level. The other input to the OR gate **1030**, in one embodiment, comes from a thermal sensor or hardware device responsible for monitoring the thermal conditions of the memory system. In one embodiment, the output signal of the OR gate **1030** is used to indicate to the pool manager **1020** that a thermal overload condition has been detected. In response to the output signal of the OR gate **1030**, the pool manager **1020** initiates a request to the DRAM controller **1040** to resize the pools as described above, using the safe size values stored in the register **1010**.

FIG. **11** shows a block diagram of one embodiment of an apparatus for dynamically changing the sizes of the power-control pools in response to a resize request from a processor (illustrated in FIG. **9**) or in response to a hardwaredetected signal indicating a thermal overload condition (as shown in FIG. **10**). In this embodiment, either a resize request received from the processor **1110** or a signal from the OR gate **1130** will trigger the pool manager **1120** to generate a request to the DRAM controller **1140** to resize the pools using the modified refresh process described above. As explained above, if the processor initiates the request to resize, the new size values can be supplied by the processor **1110** at the time of the request. The new sizes values can also be pre-stored and determined by the pool manager **1120** based upon the type of the request received from the processor **1110**. If the output of the OR gate **1130** signals a thermal overload condition, the safe size values to be used can be retrieved from the register **1105**. In one embodiment, the register **1105** can be programmed by system software. Thus, in this configuration, the pool manager **1120** can dynamically change the sizes of the pools in response to a resize request from the processor **1110** or in response to a signal from the OR gate **1130** indicating a thermal overload condition.

FIG. **12** illustrates a flow diagram of one embodiment of a method for dynamically changing the sizes of the power-control pools in response to a request to resize from a processor or other system units. The method **1200** starts at block **1201** and proceeds to loop **1205**. At loop **1205**, a pool manager awaits for a request to resize the pools from the processor. Upon receiving a request to resize from the processor, the pool manager proceeds out of the loop **1205** to initiate a request to a DRAM controller to resize the pools at block **1209**. At loop **1213**, the DRAM controller proceeds out of the wait loop **1213** when the time to refresh the memory devices has come. At block **1217**, in response to the resize request generated by the pool manager, a modified refresh process as described above is performed to place all memory devices in a specific operating mode or power state (e.g., nap mode) at the end of the refresh. In one embodiment, the modified refresh process includes activating all memory devices, refreshing all memory devices, and napping down all memory devices after they have been refreshed. At block **1221**, the pools are reinitialized by the pool manager upon receiving a signal indicating that all memory devices have been placed in a known, specific operating mode or power state (e.g., nap mode). The method then proceeds to end at block **1291**.

FIG. **13** illustrates a flow diagram of one embodiment of a method for dynamically changing the sizes of the power-control pools in response to system events without software intervention. The method **1300** starts at block **1301** and enters a wait state at loop **1305**. If a signal indicating a thermal overload condition is detected, the method then proceeds out of loop **1305** to generate a request to resize the pools to safe size values at block **1309**. At loop **1313**, the DRAM controller proceeds out of the wait loop **1313** when the time to refresh the memory devices has come. At block **1317**, in response to the resize request generated by the pool manager, a modified refresh process as described above is performed to place all memory devices in a specific operating mode or power state (e.g., nap mode) at the end of the refresh. In one embodiment, the modified refresh process

includes activating all memory devices, refreshing all memory devices, and napping down all memory devices after they have been refreshed. At block **1331**, upon receiving a signal indicating that all memory devices have been placed in a known, specific operating mode or power state (e.g., nap mode), the pools are reinitialized by the pool manager using safe size values corresponding to the detected thermal overload condition. As described above, in one embodiment, the safe size values to be used in response to a thermal overload condition or variations thereof can be programmed by system software and stored in a register accessible by the pool manager. The method then proceeds to end at block **1391**.

FIG. **14** shows a flow diagram of one embodiment of a method for dynamically changing the sizes of the power-control pools in response to a request to resize received from a processor (described above with respect to FIG. **12**) or in response to a hardware-detected signal indicating a thermal overload condition (as described above with respect to FIG. **13**). The method **1400** starts at block **1401** and proceeds to decision block **1405**. At decision block **1405**, the method proceeds to decision block **1409** if no thermal overload condition is detected. Otherwise the method proceeds to block **1413**. At decision block **1409**, the method loops back to decision block **1405** if there is no resize request from the processor. Otherwise the method proceeds to block **1413**. At block **1413**, the pool manager initiates a request to resize. At loop **1417**, the DRAM controller proceeds out of the wait loop **1417** when the time to refresh the memory devices has come. At block **1421**, in response to the resize request generated by the pool manager, a modified refresh process as described above is performed to place all memory devices in a specific operating mode or power state (e.g., nap mode) at the end of the refresh. In one embodiment, the modified refresh process includes activating all memory devices, refreshing all memory devices, and napping down all memory devices after they have been refreshed. At block **1431**, upon receiving a signal indicating that all memory devices have been placed in a known, specific operating mode or power state (e.g., nap mode), the pools are reinitialized by the pool manager using the new size values corresponding to the condition that triggers the resize request. If the resize request is triggered by the processor, the new size values to be used can be either supplied by the processor when the processor initiates the request or pre-stored and determined by the pool manager based on the request type received from the processor. If the resize request is triggered by the signal indicating a thermal overload condition, safe size values corresponding to the detected thermal overload condition are used by the pool manager to reinitialize the pools. As described above, in one embodiment, the safe size values to be used in response to a thermal overload condition or variations thereof can be programmed by system software and stored in a register accessible by the pool manager. The method then proceeds to end at block **1491**.

The invention has been described in conjunction with the preferred embodiment. It is evident that numerous alternatives, modifications, variations and uses will be apparent to those skilled in the art in light of the foregoing description.

What is claimed is:

1. A method of dynamically changing the size of a power control pool that is used to manage the power consumption of memory devices in a memory system, the method comprising:

receiving a request to change the size of the power control pool;

placing the memory devices in a specific operating mode after they are refreshed in a periodic refresh cycle; and

changing the size of the power control pool based upon a new size value associated with the request in response to a signal indicating that the memory devices have been placed in the specific operating mode.

2. The method of claim **1** wherein the request to change the size of the power control pool is initiated by a processor unit.

3. The method of claim **2** wherein the new size value is supplied by the processor with the request.

4. The method of claim **2** wherein the new size value is pre-stored.

5. The method of claim **1** wherein the request to change the size of the power control pool is initiated by a signal generated by a control device.

6. The method of claim **5** wherein the signal generated by the control device is used to indicate a thermal condition that exceeds a predetermined threshold value.

7. The method of claim **5** wherein the control device is a thermal sensor monitoring the thermal condition of the memory devices.

8. The method of claim **5** wherein the control device is a thermal sensor monitoring the thermal condition of other devices.

9. The method of claim **5** wherein the new size value is maintained in a storage unit.

10. The method of claim **9** wherein the storage unit comprises one or more registers.

11. The method of claim **1** wherein receiving comprises:

receiving the request to change the size of the power control pool from a processor.

12. The method of claim **1** wherein receiving comprises:

detecting a signal from a thermal control unit indicating a thermal overload condition outside of the memory system.

13. The method of claim **1** wherein receiving comprises:

detecting a signal from a thermal control unit indicating that the temperature of the memory system has exceeded a threshold level.

14. The method of claim **1** wherein placing the memory devices in the specific operating mode comprises:

performing a periodic refresh cycle including:

activating the memory devices;

refreshing the memory devices after the memory devices have been activated; and

placing the memory devices in the specific operating mode after the memory devices have been refreshed.

15. The method of claim **1** wherein changing the size the power control pool comprises:

updating the power control pool to reflect that the memory devices have been placed in the specific operating mode; and

changing the size of the power control pool to the new size value associated with the request received.

16. The method of claim **1** wherein the specific operating mode is a nap mode.

17. An apparatus for dynamically changing the size of a power control pool that is used to manage the power consumption of memory devices in a memory system, the memory devices having active and inactive operating modes, the apparatus comprising:

receive logic to receive a request to change the size of the power control pool;

special refresh logic to place the memory devices in the inactive operating mode after they are refreshed in a

periodic refresh cycle, in response to the request to change the size of the power control pool; and

update logic to change the size of the power control pool to a new size value associated with the request after the memory devices have been refreshed and placed in the inactive operating mode.

18. The apparatus of claim 17 wherein the request to change the size of the power control pool is initiated by a processor unit.

19. The apparatus of claim 18 wherein the signal generated by the control device is used to indicate a thermal condition exceeding a predetermined threshold value.

20. The apparatus of claim 18 wherein the receive logic comprises:

logic to receive the request to change size from the processor.

21. The apparatus of claim 18 wherein the new size value is supplied by the processor with the request.

22. The apparatus of claim 18 wherein the new size value is pre-stored.

23. The apparatus of claim 17 wherein the request to change the size of the power control pool is initiated by a signal generated by a control device.

24. The apparatus of claim 23 wherein the control device is a thermal sensor monitoring the thermal condition of the memory devices.

25. The apparatus of claim 23 wherein the control device is a thermal sensor monitoring the thermal condition of other devices.

26. The apparatus of claim 23 wherein the receive logic comprises:

logic to detect the signal generated by the control device.

27. The apparatus of claim 23 wherein the new size value is maintained in a storage unit.

28. The apparatus of claim 27 wherein the storage unit comprises one or more registers.

29. The apparatus of claim 17 wherein the special refresh logic comprises:

activation logic to place the memory devices in the active operating mode;

logic to refresh the memory devices after they have been activated; and

logic to place the memory devices in the inactive operating mode after they have been refreshed.

30. The apparatus of claim 17 wherein the update logic comprises:

logic to update the power control pool to reflect that the memory devices have been placed in the inactive operating mode; and

logic to change the size of the power control pool to the value associated with the request received.

31. The apparatus of claim 17 wherein the inactive operating mode is a nap mode.

32. An apparatus for dynamically changing the size of a power control pool that is used to manage the power consumption of memory devices in a memory system, the method comprising:

means for receiving a request to change the size of the power control pool;

means for placing the memory devices in a specific operating mode after they are refreshed in a periodic refresh cycle; and

means for changing the size of the power control pool based upon a new size value associated with the request in response to a signal indicating that the memory devices have been placed in the specific operating mode.

33. The apparatus of claim 32 wherein the request to change the size of the power control pool is initiated by a processor unit.

34. The apparatus of claim 32 wherein the request to change the size of the power control pool is initiated by a signal generated by a control device.

35. The apparatus of claim 34 wherein the signal generated by the control device is used to indicate a thermal condition that exceeds a predetermined threshold value.

36. The apparatus of claim 34 wherein means for receiving comprises:

means for detecting the signal generated by the control device.

37. The apparatus of claim 32 wherein means for placing the memory devices in the specific operating mode comprises:

means for activating the memory devices; and

means for refreshing the memory devices after the memory devices have been activated.

38. The apparatus of claim 32 wherein means for changing the size the power control pool comprises:

means for updating the power control pool to reflect that the memory devices have been placed in the specific operating mode; and

means for changing the size of the power control pool to the new size value associated with the request received.

39. An apparatus for dynamically balancing the performance and power consumption levels of a memory system containing multiple dynamic random access memory (DRAM) devices, the multiple DRAM devices having at least two different first and second operating modes, the first operating mode corresponding to a higher level of performance and power consumption than the second operating mode, the apparatus comprising:

logic to maintain a first list based upon a first number, the first list indicating which DRAM devices being in the first operating mode, the first number indicating the maximum number of DRAM devices allowed to be in the first operating mode;

logic to receive a request to change the first number to a new value; and

logic to update the first list and the first number in response to the request to change the first number.

40. The apparatus of claim 39 wherein the logic to update the first list and the first number comprises:

logic to request the multiple DRAM devices to be placed in the second operating mode after being refreshed; and

logic to reinitialize the first list after the multiple DRAM devices being placed in the second operating mode.

41. The apparatus of claim 39 further comprising:

refresh logic to refresh the multiple DRAM devices periodically.

42. The apparatus of claim 39 wherein the first operating mode is an active mode and the second operating mode is a standby mode.

43. The apparatus of claim 39 wherein the first operating mode is an active mode and the second operating mode is a nap mode.

44. The apparatus of claim 39 wherein the first operating mode is a standby mode and the second operating mode is a nap mode.

45. A memory controller for dynamically balancing the performance and power consumption levels of a memory system containing multiple dynamic random access memory (DRAM) devices, the multiple DRAM devices having at

least two different first and second operating modes, the first operating mode corresponding to a higher level of performance and power consumption than the second operating mode, the memory controller comprising:

refresh logic to refresh the multiple DRAM devices periodically; and

a pool manager to control the performance and power consumption levels of the multiple DRAM devices comprising:

logic to maintain a first list based upon a first number, the first list indicating which DRAM devices being in the first operating mode, the first number indicating the maximum number of DRAM devices allowed to be in the first operating mode;

logic to receive a request to change the first number to a new value; and

logic to update the first list and the first number in response to the request to change the first number.

46. The memory controller of claim **45** wherein the logic to update the first list and the first number comprises:

logic to request the multiple DRAM devices to be placed in the second operating mode after being refreshed; and

logic to reinitialize the first list after the multiple DRAM devices being placed in the second operating mode.

47. A system for dynamically balancing the performance and power consumption levels of a memory system containing multiple dynamic random access memory (DRAM) devices, the multiple DRAM devices having at least two different first and second operating modes, the first operating mode corresponding to a higher level of performance and power consumption than the second operating mode, the system comprising:

a processor; and

a memory controller coupled to the processor, the memory controller comprising:

refresh logic to refresh the multiple DRAM devices periodically; and

a pool manager to control the performance and power consumption levels of the multiple DRAM devices comprising:

logic to maintain a first list based upon a first number, the first list indicating which DRAM devices being in the first operating mode, the first number indicating the maximum number of DRAM devices allowed to be in the first operating mode;

logic to receive a request to change the first number to a new value; and

logic to update the first list and the first number in response to the request to change the first number.

48. The system of claim **47** wherein the logic to update the first list and the first number comprises:

logic to request the multiple DRAM devices to be placed in the second operating mode after being refreshed; and

logic to reinitialize the first list after the multiple DRAM devices being placed in the second operating mode.

49. A system for dynamically balancing the performance and power consumption levels of a memory system containing multiple dynamic random access memory (DRAM) devices, the multiple DRAM devices having at least two different first and second operating modes, the first operating mode corresponding to a higher level of performance and power consumption than the second operating mode, the system comprising:

a processor to initiate a request to change the number of DRAM devices allowed in the first operating mode;

a thermal control unit to generate a signal indicating a thermal condition exceeding a thermal threshold value; and

a memory controller coupled to the processor and the thermal control unit, the memory controller comprising:

refresh logic to refresh the multiple DRAM devices periodically; and

a pool manager to control the performance and power consumption levels of the multiple DRAM devices comprising:

logic to maintain a first list based upon a first number, the first list indicating which DRAM devices being in the first operating mode, the first number indicating the maximum number of DRAM devices allowed to be in the first operating mode;

logic to receive a request to change the first number to a new value;

logic to detect the signal indicating the thermal condition; and

logic to update the first list and the first number in response to the request from the processor or the signal indicating the thermal condition.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,330,639 B1
DATED : December 11, 2001
INVENTOR(S) : Fanning et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 5,
Line 35, before "control unit 131', delete "10", insert -- I/O --.

Column 7,
Line 5, delete "modem", insert -- modern --.

Signed and Sealed this

Nineteenth Day of November, 2002

Attest:

Attesting Officer

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US006546343B1

(12) **United States Patent**
Batra et al.

(10) **Patent No.: US 6,546,343 B1**
(45) **Date of Patent: Apr. 8, 2003**

(54) **BUS LINE CURRENT CALIBRATION**

(75) Inventors: **Pradeep Batra**, Santa Clara, CA (US); **Rick A. Rutkowski**, Sunnyvale, CA (US)

(73) Assignee: **Rambus, Inc.**, Los Altos, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 199 days.

(21) Appl. No.: **09/711,606**

(22) Filed: **Nov. 13, 2000**

(51) **Int. Cl.**[7] **H03B 1/00**; H04B 17/00

(52) **U.S. Cl.** **702/64**; 327/108; 326/62; 326/80

(58) **Field of Search** 702/107, 64; 327/108; 326/62, 80; 710/1, 25, 36, 58

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,094,075	A *	7/2000	Garrett et al.	327/108
6,160,851	A *	12/2000	Brown et al.	333/177
6,294,934	B1 *	9/2001	Garrett et al.	327/108
6,313,670	B1 *	11/2001	Song et al.	327/108
6,313,776	B1 *	11/2001	Brown	341/144
6,330,194	B1 *	12/2001	Thomann et al.	365/189.05
6,333,639	B1 *	12/2001	Lee	326/30
2002/0050844	A1 *	5/2002	Lau et al.	327/108
2002/0087280	A1 *	7/2002	To et al.	702/107

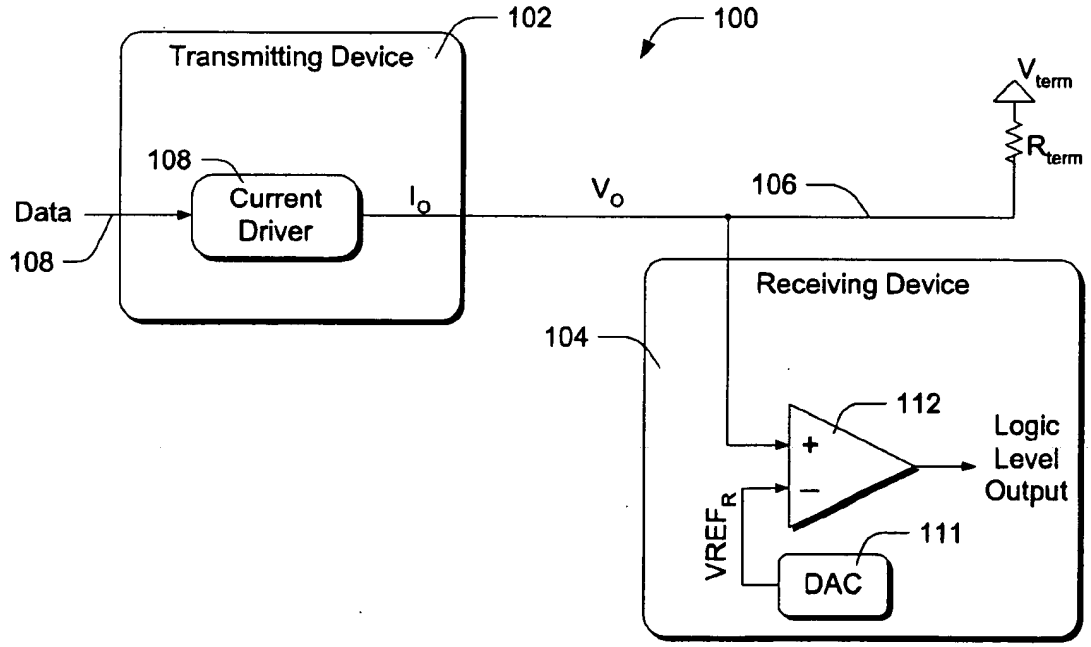
* cited by examiner

Primary Examiner—Patrick Assouad
(74) *Attorney, Agent, or Firm*—Lee & Hayes, PLLC

(57) **ABSTRACT**

Disclosed herein is a method and system for calibrating line drive currents in systems that generate data signals by varying line drive currents and that interpret the data signals by comparing them to one or more reference voltages. The calibration includes varying the line drive current at a transmitting component. At different line drive currents, a receiver reference voltage is varied while the transmitting component transmits data to a receiving component. At each line drive current, the system records the highest and lowest receiver reference voltages at which data errors do not occur. The system then examines the recorded high and low receiver reference voltages to determine a desirable line drive current.

45 Claims, 5 Drawing Sheets



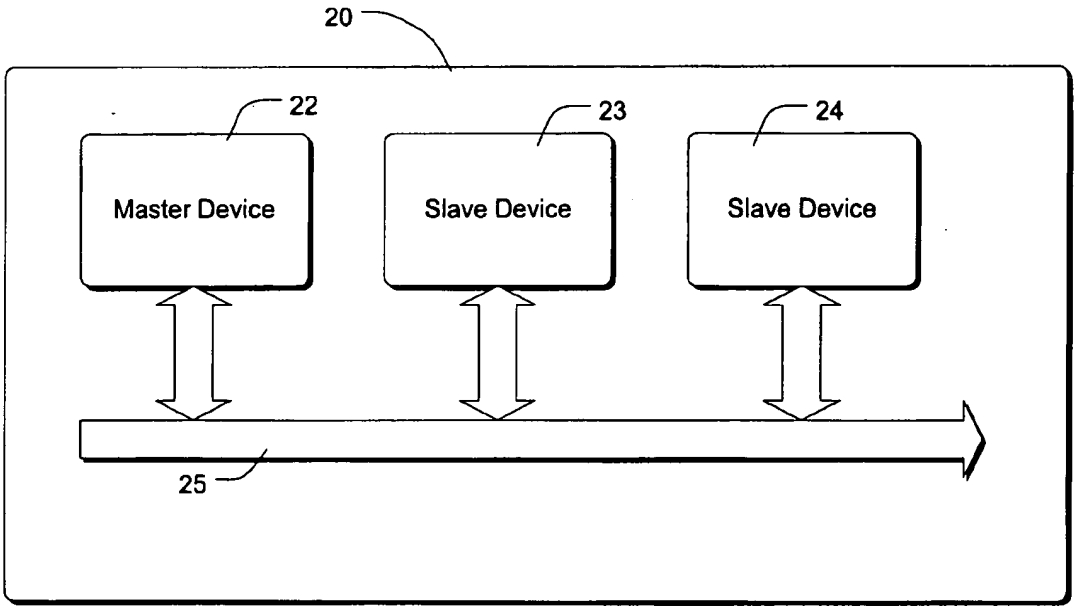


Fig. 1
Prior Art

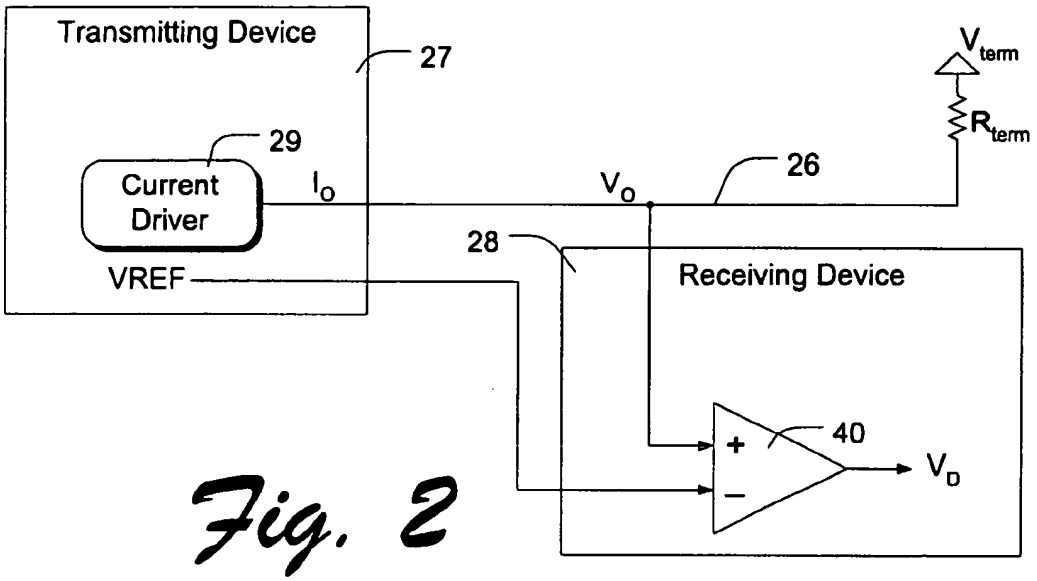


Fig. 2
Prior Art

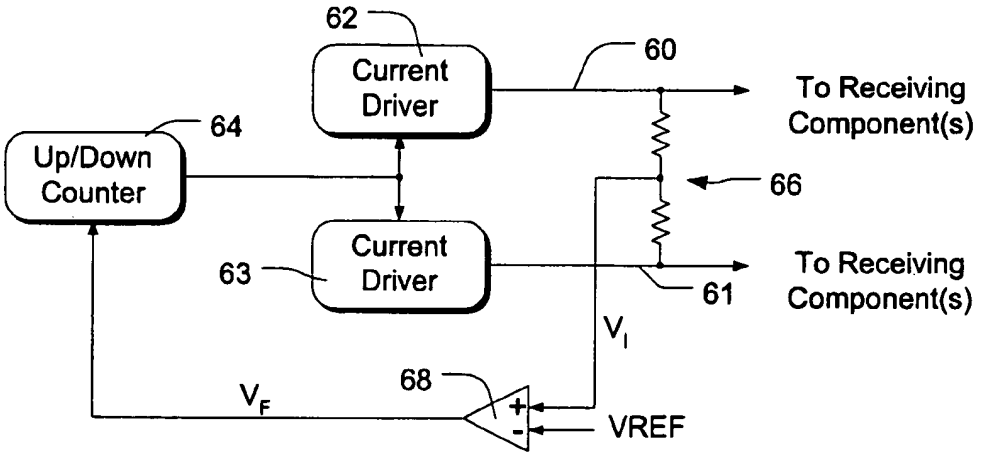


Fig. 3
Prior Art

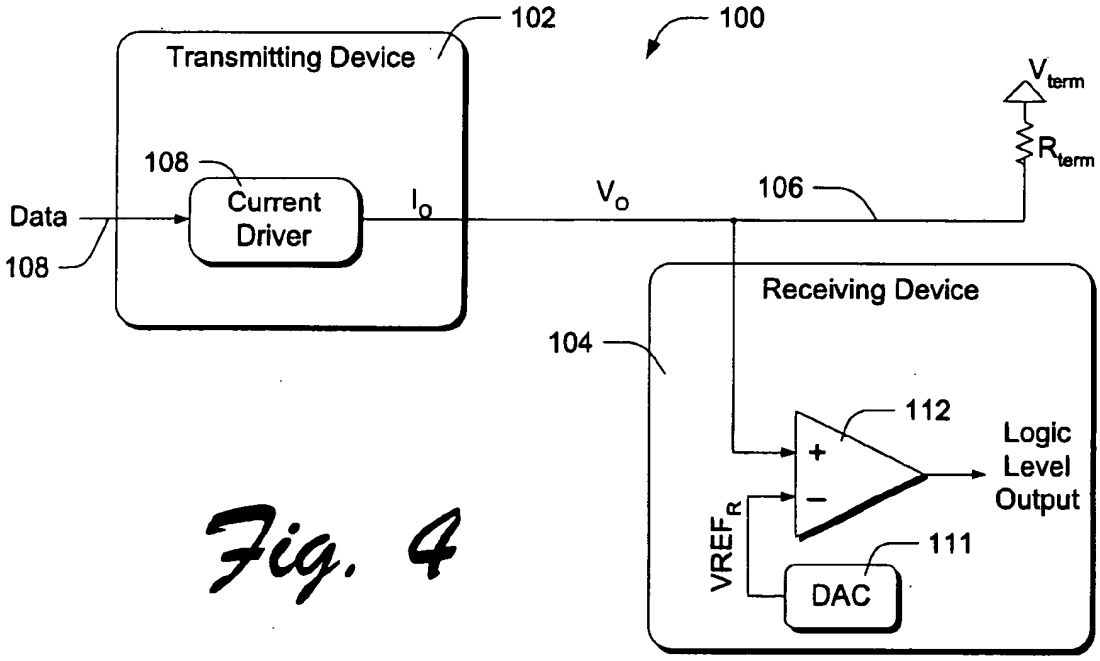

102
100
Transmitting Device
V_{term}
R_{term}
108
Current Driver
Data
108
I_o
V_o
106
Receiving Device
104
112
Logic Level Output
$VREF_R$
111
DAC

Fig. 4

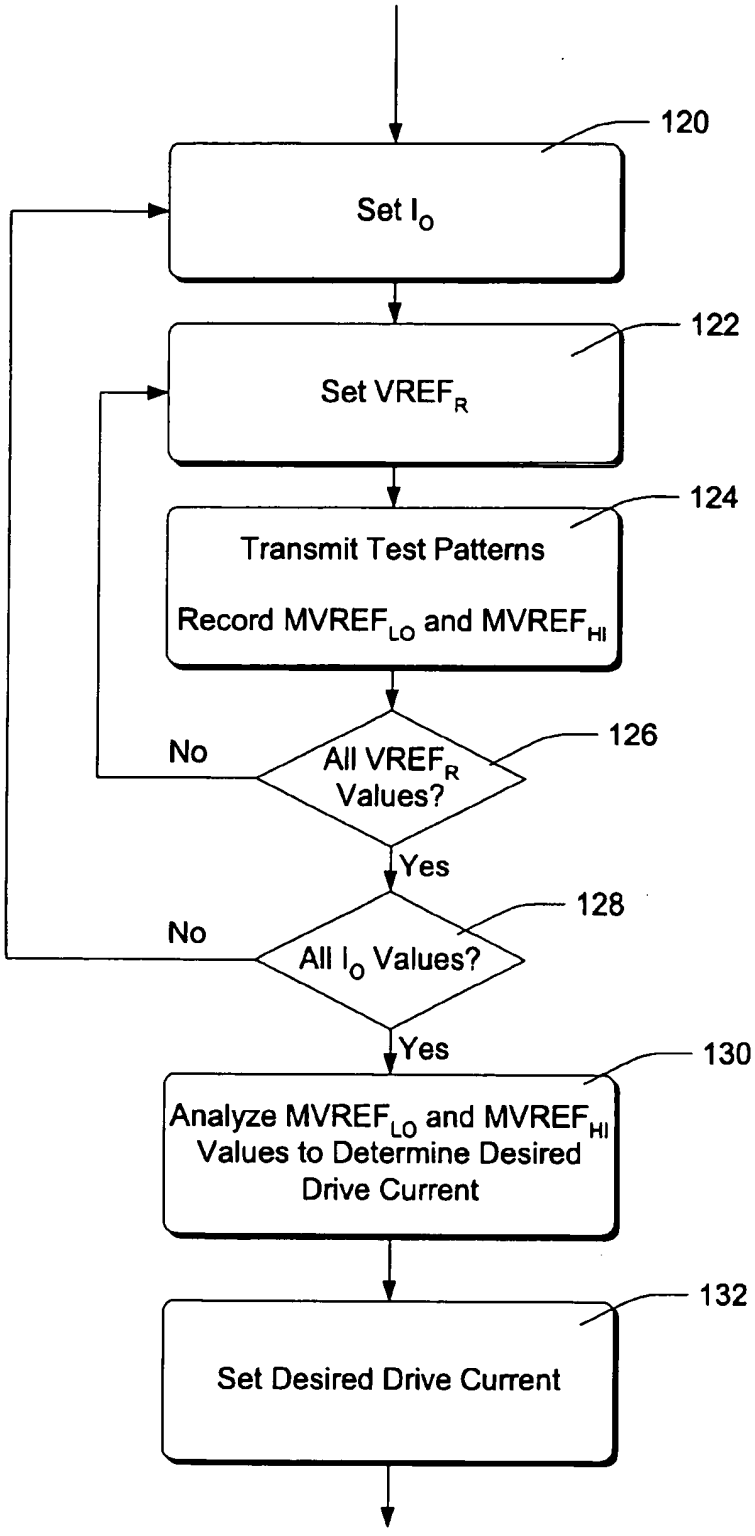

120
Set I_O
122
Set $VREF_R$
124
Transmit Test Patterns
Record $MVREF_{LO}$ and $MVREF_{HI}$
126
All $VREF_R$ Values?
No
Yes
128
All I_O Values?
No
Yes
130
Analyze $MVREF_{LO}$ and $MVREF_{HI}$ Values to Determine Desired Drive Current
132
Set Desired Drive Current

Fig. 5

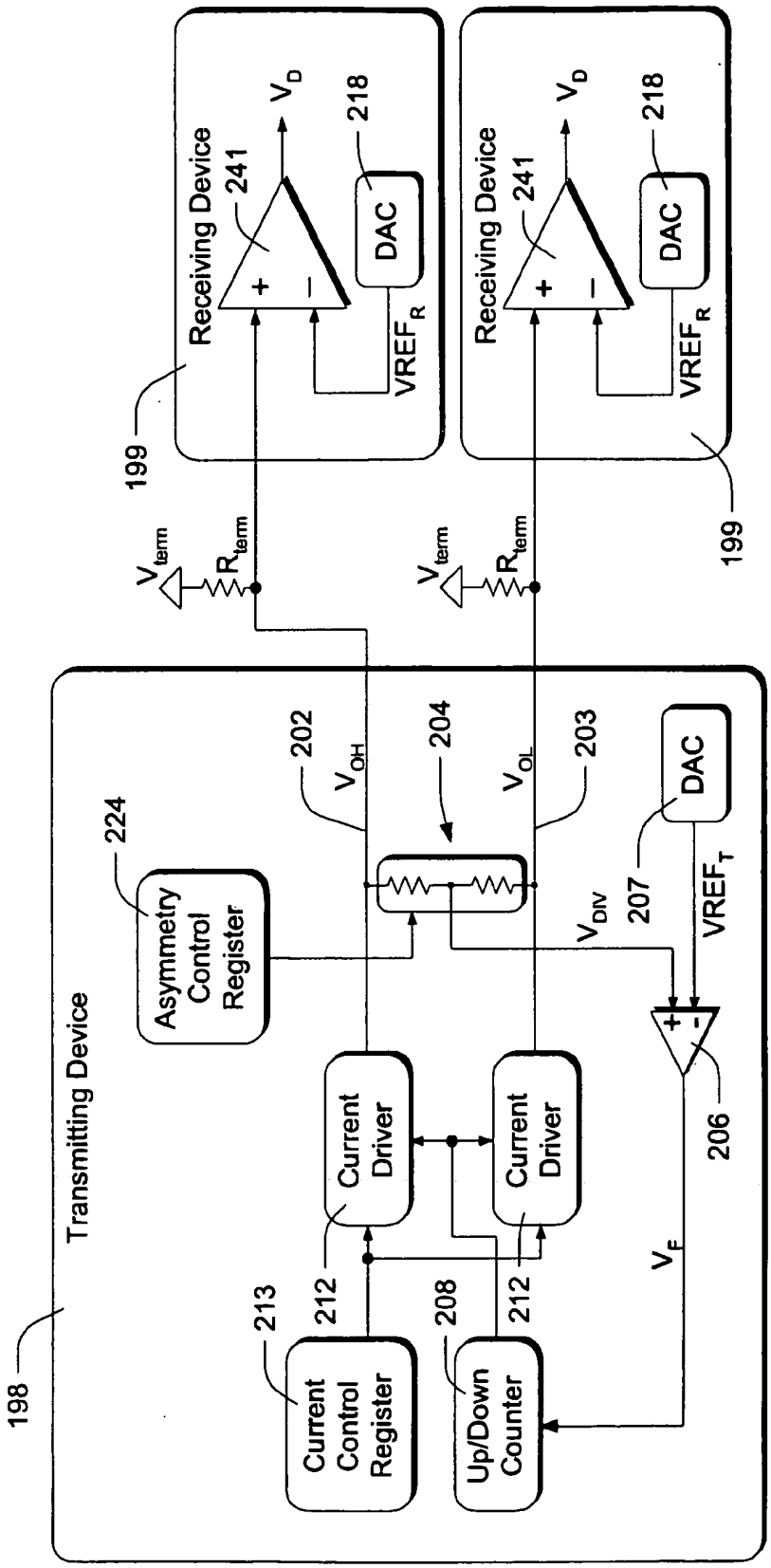

Receiving Device
241
V_D
218
DAC
$VREF_R$
199
V_{term}
R_{term}
Transmitting Device
198
224
Asymmetry Control Register
202
V_{OH}
204
V_{OL}
203
V_{DIV}
207
DAC
$VREF_T$
206
V_F
Current Driver
Current Driver
213
212
208
212
Current Control Register
Up/Down Counter

Fig. 6

BUS LINE CURRENT CALIBRATION

TECHNICAL FIELD

This invention relates to bus systems in which line voltages are generated by varying line currents and are interpreted with reference to a reference voltage.

BACKGROUND OF THE INVENTION

FIG. **1** is a block diagram showing a high-speed digital computer bus system **20**. The bus system includes a number of discrete devices **22–24**, which communicate over an electrical bus **25** at very high speeds. The bus includes a plurality of data transmission lines.

This system includes a master device **22** and a plurality of slave devices **23–24**. The master device **22** initiates and controls data exchanges over bus **25**. During a data exchange, any one of devices **22–24** can act as either a transmitting component or a receiving component. Generally, there is only one transmitting component during any single data exchange. However, there can be one or a is plurality of receiving components during a data exchange.

FIG. **2** illustrates the configuration and operation of a single bus line **26** between a transmitting component **27** and a receiving component **28**. The bus line is terminated at one end to a termination voltage V_{term} through a termination impedance R_{term}. Transmitting component **27** has a line current driver **29**, which produces line voltages with specified relationships to a reference voltage VREF.

More specifically, driver **29** is a current source or sink that creates desired voltage drops across termination impedance R_{term}. The current driver **29** is turned on or otherwise enabled to produce one logic level voltage, and is turned off or otherwise disabled to produce another logic level voltage. In actual embodiment, the current driver **29** sinks current when enabled, and does not sink or source current when disabled. When disabled, the line voltage is approximately equal to V_{term}. When enabled, the line voltage is lower than V_{term}, because of a voltage drop through termination impedance R_{term}.

As an example, suppose that V_{term} is 2.5 volts. When driver **29** is disabled there is no current through the bus line, and the bus line voltage is equal to V_{term}, or 2.5 volts. This is the high logic level, and is referred to as V_{OH}. On the other hand, when driver **29** is enabled the current through the bus line drops the line voltage to a lower value V_{OL}, which in this example is 1.9 volts. V_{OL} is the low logic level.

The voltage difference between V_{OH} and V_{OL}, also referred to as a line voltage swing V_{swing}, is controlled by the value of termination resistance V_{term} and the amount of line current I_O (which is controlled by the current driver **29**). It is desirable to limit the line voltage swing as much as possible to enable higher bus speeds. If the voltage swing is too small, however, a receiving component will not be able to reliably distinguish between high and low logic level voltages.

FIG. **2** also illustrates how the line voltage is interpreted at receiving component or device **28**. Specifically, the received line voltage V_O is compared to reference voltage VREF by a comparator **40**. If V_O is greater than VREF, the line voltage represents a high logic level. If V_O is less than VREF, the line voltage represents a low logic level.

For this determination to be valid, the transmitting component needs to set its V_{OH} and V_{OL} relative to VREF. Preferably, V_{OH} and V_{OL} are established symmetrically around VREF. In the example of FIG. **3**, V_{OH} is 2.5 volts, V_{REF} is 2.2 volts, and V_{OL} is 1.9 volts. This yields a 0.6 volt voltage swing: 0.3 volts on either side of VREF.

FIG. **3** shows a circuit for creating a symmetrical voltage swing around VREF during a calibration process. This circuit, which is used only during the calibration, utilizes two different bus lines **60** and **61**, each of which are similar to the bus line **26** shown in FIG. **2**.

The calibration circuit has current drivers **62** and **63**, and a current control **64** which in this case is an up/down counter. Current drivers **62** and **63** are switched on and off by data control lines (not shown) to create high and low voltage levels V_{OH} and V_{OL} on the corresponding bus lines. When a driver is on, the magnitude of its output current is determined by the value contained in up/down counter **64**.

Bus lines **60** and **61** extend to receiving components and a termination resistor (not shown). Within the transmitting component, however, the high and low output voltages V_{OH} and V_{OL} are sampled for purposes of adjusting the current driver outputs to create a symmetric voltage swing. Specifically, a simple R over R resistive voltage divider **66** is placed between a line producing a high logic voltage V_{OH} and another line producing a low logic level V_{OL}. In this case, it is assumed that line **60** is at the high voltage level, with current driver **62** inactive; and bus line **61** is at the low voltage level, with current driver **63** being active. Furthermore, the resistive divider **66** is configured to produce an intermediate output voltage V_I that is equal to $(V_{OH}+V_{OL})/2$. For symmetry around VREF, V_I should be equal to VREF. A feedback system is used to minimize the voltage difference between V_I and VREF. Both V_I and VREF are connected to the inputs of a comparator **68**, which produces a logic voltage V_F that is high when V_I–VREF>0, and low when V_I–VREF<0. V_F is then connected to counter **64**. The output of the counter, in turn, is connected to control the output of current drivers **62** and **63**.

The circuit works as follows. During calibration, counter **64** is enabled and/or clocked, and repetitively adjusts its output either up or down depending on the logic value of V_F. This increases or decreases the output of current driver **63**. The output current is thus adjusted until the value of counter **64** has settled. At this point, V_I–VREF=zero—meaning that V_I=VREF and that V_{OH} and V_{OL} are symmetric around VREF. At this point, the value of counter **64** is frozen until the next calibration (although minor adjustments might be made by temperature control circuits).

In most cases, this calibration is performed at system initialization. Optionally, the calibrated current control value (from the counter) can be stored in a current control register and used during normal bus operation to control the magnitude of I_O. This value can then be subject to temperature correction circuits to determine the current control value at any given time. Alternatively, the calibration can be performed periodically to account for temperature and voltage variations.

Ideally, both the transmitting component and a receiving component have the same value of VREF. In practice, however, this can be difficult to achieve due to signal line losses and/or noise. Accordingly, VREF at the receiving component is often somewhat different than VREF at the transmitting component. Furthermore, V_{OH} and V_{OL} often change as they propagate through the signal line, again due to losses and noise. Thus, the relationship between V_{OH}, V_{OL} and VREF may not be the same at the transmitting component as it is at the receiving component. In other words, V_{swing} might not be symmetric around VREF by the time the signals reach a receiving component.

In the bus configuration described above, line losses generally affect V_{OL} more than V_{OH}. At V_{OL}, the voltage is being produced by a current through the bus line, so the voltage can be affected along the length of the bus line by resistive and capacitive loads. At V_{OH}, however, there is no line current, and therefore less opportunity for the voltage to be affected along the length of the bus line. This situation affects both the line voltage swing and the relationship of V_{OL} with VREF.

The non-symmetry at the receiving component has negative effects. If V_{OL} **11** is higher at the receiving component, the voltage margin from VREF to V_{OL} is decreased. When V_{OL} is lower at the receiving component, low-side margin is increased, but the higher V_{swing} would cause more reflections, which could degrade the high-side margin during a subsequent data transfer cycle

This issue has been addressed by introducing a degree of asymmetry at the transmitting component in order to provide symmetry at the receiving component: the current drivers at the transmitting component are adjusted to achieve voltage symmetry at the receiving component. The amount of asymmetry at the transmitting component is referred to as the overdrive factor (ODF).

A desired asymmetry at the transmitting component can be created by varying the ratio of voltage divider **66**. Thus, instead of producing a signal V_t that is 50% of the way from V_{OL} to V_{OH}, the resistors can be chosen to implement any other percentage. This creates asymmetry at the transmitting component to correct for any asymmetry that would otherwise be present at the receiving component.

In actual embodiment, the ratio of voltage divider **66** has been controlled by a symmetry control register. Different values can be loaded into the symmetry control register to create different degrees of asymmetry at the transmitting component. Symmetry control values can be stored for a plurality of different receivers, and used when transmitting to those receivers. This accounts for variations in conditions at different receivers.

In some such circuits, the value of counter **64** is stored after completion of the calibration process, and loaded into a current control register during actual operation. A plurality of values can be stored, corresponding to different receiving components. The current control register is reloaded for communication with different receiving components.

The desired line current and corresponding divider percentage or ratio are determined during system design—prior to manufacture of the transmitting component or prior to manufacture of a circuit that utilizes the transmitting component. The determination is based on testing and/or simulating, and choosing voltage divider ratios that are predicted to work with the different receivers in light of the actual circuit layout. In some cases, the transmitting component includes logic for predicting required asymmetry values based on known system parameters such as distances between components.

The inventors, however, have discovered and developed a way to dynamically determine appropriate line drive currents at system initialization, based on tested characteristics of the circuits themselves.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a high-speed digital computer bus system in accordance with the prior art.

FIG. **2** is a block diagram illustrating a bus line transmitter and receiver in accordance with the prior art.

FIG. **3** is a block diagram illustrating a prior art method of controlling bus line current.

FIG. **4** is a block diagram of a first embodiment illustrating a bus line calibration method.

FIG. **5** is a flowchart showing methodological operations performed in a bus line calibration method.

FIG. **6** is a block diagram of a second embodiment illustrating a bus line calibration method.

DETAILED DESCRIPTION

The following description sets forth specific embodiments of a bus driver calibration method and circuits having elements recited in the appended claims. The embodiment is described with specificity in order to meet statutory requirements. However, the description itself is not intended to limit the scope of this patent. Rather, the inventors have contemplated that the claimed invention might also be embodied in other ways, to include different elements or combinations of elements similar to the ones described in this document, in conjunction with other present or future technologies.

FIG. **4** shows pertinent parts of a data communications system **100** that can be used to implement a method of calibrating or finding a desirable line drive current. The system allows such calibration during device operation. The calibration is usually performed at system startup or initialization. Through actual testing during the calibration procedure, asymmetry values are determined for different transmitting components. Furthermore, each transmitting component can find a different asymmetry value for every possible receiving component. This allows an optimal line current to be used for each different receiver.

The overall system is of a type that interprets data signals by comparing them to reference voltages. Signals are transmitted over a plurality of data signaling lines, also referred to as bus lines. Each bus line is terminated to a termination voltage V_{term}, through a termination resistance R_{term}. The data signals comprise line voltages that are produced by line drive currents through R_{term}. The line voltages are compared to the reference voltages to determine whether the voltages represent high or low logic values.

This type of bus system is used in various environments. One of its most common uses is for high-speed data communications busses between microprocessors and peripheral devices such as high-speed memory devices. Such bus systems are also used in high-speed memory subsystems, between memory devices and memory device controllers.

For purposes of discussion, the illustration of FIG. **4** has been greatly simplified, and shows only parts of the transmitting and receiving components that might be used in a calibration process.

FIG. **4** shows a single transmitting component **102** and a single receiving component **104**. A data bus line **106** extends between transmitting component **102** and receiving component **104**. In practice, there are a plurality of bus lines, and each bus line might be connected to one or more transmitting components and/or one or more receiving components.

Within transmitting component **102**, a current driver **108** generates a current I_O though bus line **106**. The current driver is responsive to a logic level data signal **110** to switch the current driver on or off, generating a signal voltage V_O. A high signal voltage is referred to as V_{OH}, and is the result of switching the driver off. A low signal voltage is referred to as V_{OL}, and is the result of switching the driver on. V_{OH} and V_{OL} are usually used to represent high and low binary logic levels "1" and "0".

Receiving component **104** has a reference voltage $VREF_R$ that can be independent of any voltage in transmitting component **102**. $VREF_R$ is generated by a digital-to-analog converter (DAC) **111**. The DAC can be integrated with receiving component **104** or it can be separate from receiving component **104**.

Receiver **104** includes a voltage comparator **112** that compares two analog voltage inputs and produces a binary voltage signal indicating which of the two input voltages is greater in magnitude. A comparator such as this is associated with each bus line to evaluate whether a logic signal represents a high or low logic voltage level.

One input of comparator **112** receives receiver reference voltage $VREF_R$. The other input of comparator **112** is connected to bus line **106**. In response to these two inputs, the comparator generates a logic level signal or voltage V_D, representing a data bit or other information that has been transmitted from transmitting component **102**. The value of V_D is dependent on whether V_O is greater or less than $VREF_R$.

FIG. **5** illustrates a method of calibrating current driver **108** so that the resulting voltage swing V_{swing} is symmetrical around $VREF_R$. Such a calibration is performed during system startup or during initialization of the transmitting and/or receiving components.

The method comprises varying the line drive current I_O—sequentially setting I_O to each of a range of possible values I_O. This is accomplished by sequentially configuring the transmitting component's current driver **108** to supply line drive current I_O at a plurality of discrete values $I_O[0]$ through $I_O[n]$, as indicated in FIG. **5** by blocks **120** and **128**.

Blocks **122** and **126** indicate an operation of varying the receiver reference voltage $VREF_R$ through a predetermined sequence of different available voltages. This operation is repeated for every value of I_O.

Operation **124** is performed for each $VREF_R$ value, and comprises transmitting data from transmitting component **102** to receiving component **104** to find high and low receiver reference voltages at which data errors do not occur.

More specifically, operations **122**, **124**, and **126** comprise setting $VREF_R$ to some intermediate value and then varying $VREF_R$ downward. At each value of $VREF_R$, the transmitting component sends worst-case test data and then determines whether or not there was an error in receiving the data at the receiving component. Initially, at intermediate voltages, errors are unlikely. However, as $VREF_R$ is varied further downward, the margin between $VREF_R$ and V_{OL} will become small enough to cause data errors. This establishes the low receiver reference voltage $MVREF_{LO}$—the lowest $VREF_R$ at which data errors do not occur. A similar procedure, varying $VREF_R$ in the upward direction, establishes $MVREF_{HI}$—the highest $VREF_R$ at which data errors do not occur.

$MVREF_{LO}$ and $MVREF_{HI}$ are noted or recorded in arrays $MVREF_{LO}[i]$ and $MVREF_{HI}[i]$, corresponding to each drive current value $I_O[i]$.

An operation **130** comprises examining or evaluating the noted high and low receiver reference voltages $MVREF_{LO}[i]$ and $MVREF_{HI}[i]$ corresponding to different drive currents $I_O[i]$, to determine a desirable line drive current. More specifically, this operation comprises finding a pair of receiver reference voltages $MVREF_{LO}[i]$ and $MVREF_{HI}[i]$, for each value of i through n, that meet pre-specified criteria or are within specified parameters. Such predefined criteria, for example, might require that $MVREF_{LO}[i]$ is less than some specified value and that $MVREF_{HI}[i]$ is above some other specified value. Alternatively, the predefined criteria might specify that the desired drive current $I_O[i]$ is obtained when V_{swing}, the difference between $MVREF_{HI}[i]$ and $MVREF_{LO}[i]$, is at its greatest. This relationship produces the highest useful voltage margin at V_{OL}. Another criteria might specify the drive current $I_O[i]$ corresponding to the pair of $MVREF_{LO}[i]$ and $MVREF_{HI}[i]$ whose average $(MVREF_{LO}[i]-MVREF_{HI}[i])/2)$ is closest to the $VREF_R$ value that will be used during normal bus operation. This ensures that V_{OL} and V_{OH} are symmetrical around $VREF_R$ during normal bus operation.

Operation **132** comprises setting current driver **108** to the drive current $I_O[i]$ corresponding to the pair of receiver reference voltages $MVREF_{LO}[i]$ and $MVREF_{HI}[i]$ that were found in operation **130**. This drive current $I_O[i]$ is used on all of the data signaling lines originating from transmitting component **102**.

FIG. **6** shows a different embodiment of a system in which V_{OL} and V_{OH} are calibrated to provide symmetry at a receiving component. FIG. **6** shows only the current control and calibration circuitry in a transmitting component **198** and the comparison circuitry of a receiving component **199**.

The current control and calibration circuitry of transmitting component **198** uses two different bus lines **202** and **203**. A variable voltage divider **204** is located between the two bus lines. Divider **204** divides the voltage differential between a high line voltage V_{OH} and low line voltage V_{OL} to obtain an intermediate voltage V_{DIV} that is a dynamically programmable fraction between the high and low line voltages V_{OH} and V_{OL}. The output voltage V_{DIV} of variable voltage divider **204** is controlled by a factor x stored in a symmetry control register **224**: $V_{DIV}=V_{OL}+x\ (V_{OH}-V_{OL})$, where factor x is a value between 0 and 1.

The output V_{DIV} of the voltage divider is provided to a comparator **206**, along with a transmitter reference voltage $VREF_T$. $VREF_T$ is generated by a DAC **207**. The output feedback voltage V_F of the comparator is connected to control a current controller. Specifically, V_F drives the up/down count of a current control counter **208**. The value of counter **208** is received by current drivers **212** (one associated with each bus line **202** and **203**) to set or control the output line current I_O, and, consequently, the line voltage V_{OL}.

Up/down counter **208** is configured to control current drivers **212** only during a calibration operation. In normal bus operation, the current drivers' output I_O is controlled directly, by a current control register **213**.

During normal bus operation, the current drivers are independently enabled and disabled in accordance with data signals (not shown) that indicate the data values that are to be transmitted. When enabled, a bus driver produces a line current I_O according to the value held in counter **208** or current control register **213**, and a corresponding low line voltage V_{OL}. When disabled, a bus driver produces no current and a corresponding high line voltage V_{OH}, regardless of the value of counter **208** or current control register **213**.

During normal bus operation, current control register **213** remains at a fixed value except for changes to account for temperature and power supply variations. Voltage divider **204**, comparator **206**, and symmetry control register **224** are not used during normal operation.

A calibration procedure is performed to determine appropriate values for current control register **224**. During calibration, current drivers **212** are controlled by the output of up/down counter **208**, rather than by current control

register **213**. The calibration process is performed at system startup or initialization, between a transmitter and every one of its possible receivers. During this process, bus line **202** is set to its high voltage V_{OH} by disabling the associated current driver **212**. Bus line **203** is set to its low voltage V_{OL} by enabling the associated current driver **212**.

The calibration circuit of FIG. **6** regulates drive current I_O to maintain V_{DIV} approximately equal to V_{REF}. Thus, $V_{DIV}=V_{REF}$ and $V_{REF}=V_{OL}+x(V_{OH}-V_{OL})$. If x=0.5, then, V_{REF} is halfway or 50% of the way between V_{OL} and V_{OH}. If x=0.25, V_{REF} is 25% of the way from V_{OL} to V_{OH}. Thus, factor x represents the location of V_{REF} as a fraction or percentage of the voltage range between V_{OL} and V_{OH}.

In each receiving component **199**, one of the bus lines **202/203** is connected to one input of a comparator **241**. The other input of the comparator receives $VREF_R$. $VREF_T$ and $VREF_R$ are controlled independently for the transmitting component and the receiving component. Thus, while one DAC **207** generates $VREF_T$ for transmitter **198**, a different programmable DAC **218** generates $VREF_R$ for receiver **199**. In FIG. **6**, a separate DAC **218** is shown for every receiving component **199**. However, some embodiments might use a single DAC to supply a common $VREF_R$ to every receiving component.

The method of FIG. **5** is used in conjunction with the circuit of FIG. **6** to calibrate I_O. In order to vary I_O during step **120**, the factor x in symmetry control register **224** is cycled through its available values. Step **130** comprises determining the value of symmetry control register **224** that was used to produce the desired values of $MVREF_{LO}$ and $MVREF_{HI}$, and reloading this value into the symmetry control register **224** for future voltage/temperature calibrations. Step **132** comprises copying the counter value to the current control register **213** for subsequent use in normal operation of the bus lines.

During normal operation, voltage/temperature calibrations can be performed by continuously varying the value of current control register **213**. In the described embodiment, however, a voltage/temperature calibration operation is performed every 10 milliseconds using the calibration circuitry already described. This operation involves activating voltage divider **204** and up/down counter **208** to control current drivers **212**. Using the symmetry value found in step **130**, the up/down counter is allowed to settle to a new value that maintains the originally determined degree of asymmetry with regard to $VREF_T$. This new value is then loaded into current control register **213** for subsequent normal operation of the bus lines.

The circuits and methods described above allow transmitter drive currents to be optimized for each of a plurality of receivers. Furthermore, the calibration is performed dynamically, for actual pairs of transmitters and receivers. This method is more likely to produce optimum drive currents than the prior art method of predicting asymmetry values based on assumptions made during system design.

Although details of specific implementations and embodiments are described above, such details are intended to satisfy statutory disclosure obligations rather than to limit the scope of the following claims. Thus, the invention as defined by the claims is not limited to the specific features described above. Rather, the invention is claimed in any of its forms or modifications that fall within the proper scope of the appended claims, appropriately interpreted in accordance with the doctrine of equivalents.

What is claimed is:

1. In a system containing one or more data signaling lines between a transmitting component and a receiving component, a method of finding a desirable line drive current comprising:

at the transmitting component, varying the line drive current;

at different line drive currents, varying a receiver reference voltage while the transmitting component transmits data to the receiving component, to find high and low receiver reference voltages at which data errors do not occur;

noting the high and low receiver reference voltages corresponding to the different respective line drive currents; and

examining the noted high and low receiver reference voltages to find the desirable line drive current.

2. A method as recited in claim 1 wherein the examining comprises:

determining the line drive current at which the noted high and low receiver reference voltages are within specified parameters.

3. A method as recited in claim 1 wherein the examining comprises:

determining the line drive current at which the noted high and low receiver reference voltages have a specified relationship with each other.

4. A method as recited in claim 1 wherein the examining comprises:

determining the line drive current at which the noted high receiver reference voltage is greater than a first parameter and the noted low receiver reference voltage is less than a second parameter.

5. A method as recited in claim 1 wherein the examining comprises:

determining the line drive current at which the difference between the noted high and low receiver reference voltages is greatest.

6. A method as recited in claim 1 wherein the examining comprises:

determining the line drive current at which the average of the noted high and low receiver reference voltages is approximately equal to the receiver reference voltage used during normal bus line operation.

7. A method as recited in claim 1, further comprising:

using the desirable line current on said one or more data signaling lines.

8. A method as recited in claim 1, further comprising:

finding and storing desirable line drive currents for a plurality of different receiving components.

9. A method as recited in claim 1, further comprising:

finding and storing desirable line drive currents for a plurality of different receiving components; and

using the stored desirable line drive currents with different receiving components.

10. In a system containing one or more data signaling lines between a transmitting component and a receiving component, wherein an individual signal line is driven by a line drive current to produce high and low line voltages relative to a transmitter reference voltage, the high and low line voltages being distinguished with reference to a receiver reference voltage; a method of finding a desirable line drive current comprising:

at the transmitting component, enabling or disabling a desirable line drive current to produce high and low line voltages during data transfer;

calibrating the line drive current at system initialization, the calibrating comprising:

varying the line drive current to vary at least one of the high and low line voltages in relation to the transmitter reference voltage;

at different line drive currents, varying the receiver reference voltage while the transmitting component transmits data to the receiving component, to find high and low receiver reference voltages at which data errors do not occur;

noting the high and low receiver reference voltages corresponding to the different line drive currents; and

comparing the noted high and low receiver reference voltages at each line drive current to find the desirable line drive current.

11. A method as recited in claim **10** wherein the examining comprises:

determining the line drive current at which the noted high and low receiver reference voltages are within specified parameters.

12. A method as recited in claim **10** wherein the examining comprises:

determining the line drive current at which the noted high and low receiver reference voltages have a specified relationship with each other.

13. A method as recited in claim **10** wherein the examining comprises:

determining the line drive current at which the noted high receiver reference voltage is greater than a first parameter and the noted low receiver reference voltage is less than a second parameter.

14. A method as recited in claim **10** wherein the examining comprises:

determining the line drive current at which the difference between the noted high and low receiver reference voltages is greatest.

15. A method as recited in claim **10** wherein the examining comprises:

determining the line drive current at which the average of the noted high and low receiver reference voltages is approximately equal to the receiver reference voltage used during normal bus line operation.

16. A method as recited in claim **10**, further comprising:

using the desirable line current on said one or more data signaling lines.

17. A method as recited in claim **10**, further comprising:

finding and storing desirable line drive currents for a plurality of different receiving components.

18. A method as recited in claim **10**, further comprising:

finding and storing desirable line drive currents for a plurality of different receiving components; and

using the stored desirable line drive currents with different receiving components.

19. In a data communications system of a type that generates and interprets data signals by comparing them to one or more reference voltages, the data signals comprising line voltages that are produced by line drive currents; a method comprising:

dividing a voltage differential between a high line voltage and a low line voltage to obtain an intermediate voltage that is a dynamically programmable fraction between the high and low line voltages;

regulating a line drive current produced by a transmitting component to maintain the intermediate voltage approximately equal to a transmitter reference voltage;

varying the programmable fraction to vary the regulated line drive current;

at different fractions, varying a receiver reference voltage while the transmitting component transmits data to a receiving component, to find high and low receiver reference voltages at which data errors do not occur;

noting the high and low receiver reference voltages corresponding to different fractions; and

comparing the noted high and low receiver reference voltages at each line drive current to find a fraction that produces a desirable line drive current.

20. A method as recited in claim **19** wherein the examining comprises:

determining the fraction at which the noted high and low receiver reference voltages are within specified parameters.

21. A method as recited in claim **19** wherein the examining comprises:

determining the fraction at which the noted high and low receiver reference voltages have a specified relationship with each other.

22. A method as recited in claim **19** wherein the examining comprises:

determining the fraction at which the noted high receiver reference voltage is greater than a first parameter and the noted low receiver reference voltage is less than a second parameter.

23. A method as recited in claim **19** wherein the examining comprises:

determining the fraction at which the difference between the noted high and low receiver reference voltages is greatest.

24. A method as recited in claim **19** wherein the examining comprises:

determining the fraction at which the average of the noted high and low receiver reference voltages is approximately equal to the receiver reference voltage used during normal bus line operation.

25. A method as recited in claim **19**, further comprising:

using the desirable line drive current on said one or more data signaling lines.

26. A method as recited in claim **19**, further comprising:

finding and storing desirable line drive currents for a plurality of different receiving components.

27. A method as recited in claim **19**, further comprising:

finding and storing desirable line drive currents for a plurality of different receiving components; and

using the stored desirable line drive currents with different receiving components.

28. A data communications system of a type that generates and interprets data signals by comparing them to one or more reference voltages, comprising:

a transmitting component and a receiving component;

a data transmission line between the transmitting component and the receiving component;

a line current driver in the transmitting component that regulates line drive current through the data transmission line to produce high and low line voltages on the data transmission line;

control logic that performs actions comprising:

varying the regulated drive current;

at different drive currents, varying a receiver reference voltage while the transmitting component transmits data to the receiving component, to find highest and lowest receiver reference voltages at which data errors do not occur;

noting the high and low receiver reference voltages corresponding to different drive currents;

examining the noted high and low receiver reference voltages to find a drive current that produce a desirable drive current.

29. A data communications system as recited in claim **28** wherein the examining comprises:

determining the drive current at which the noted high and low receiver reference voltages are within specified parameters.

30. A data communications system as recited in claim **28** wherein the examining comprises:

determining the drive current at which the noted high and low receiver reference voltages have a specified relationship with each other.

31. A data communications system as recited in claim **28** wherein the examining comprises:

determining the drive current at which the noted high receiver reference voltage is greater than a first parameter and the noted low receiver reference voltage is less than a second parameter.

32. A data communications system as recited in claim **28** wherein the examining comprises:

determining the drive current at which the difference between the noted high and low receiver reference voltages is greatest.

33. A data communications system as recited in claim **28** wherein the examining comprises:

determining the drive current at which the average of the noted high and low receiver reference voltages is approximately equal to the receiver reference voltage used during normal bus line operation.

34. A data communications system as recited in claim **28**, the control logic performing further actions comprising:

using the desirable current on said one or more data signaling lines.

35. A data communications system as recited in claim **28**, the control logic performing further actions comprising:

finding and storing desirable drive currents for a plurality of different receiving components.

36. A data communications system as recited in claim **28**, the control logic performing further actions comprising:

finding and storing desirable drive currents for a plurality of different receiving components; and

using the stored desirable drive currents with different receiving components.

37. A data communications system of a type that generates and interprets data signals by comparing them to one or more reference voltages, comprising:

a transmitting component and a receiving component;

a data transmission line between the transmitting component and the receiving component;

a line current driver in the transmitting component that regulates line drive current through the data transmission line to produce high and low line voltages on the data transmission line;

a voltage divider in the transmitting component that produces an intermediate voltage that is a variable fraction between the high and low line voltages;

wherein the line current driver establishes the regulated current so that the intermediate voltage is approximately equal to a transmitter reference voltage;

control logic that performs actions comprising:

varying the variable fraction to vary the regulated drive current;

at different fractions, varying a receiver reference voltage while the transmitting component transmits data to the receiving component, to find high and low receiver reference voltages at which data errors do not occur;

noting the high and low receiver reference voltages corresponding to different fractions; and

comparing the noted high and low receiver reference voltages at each line drive current to find a fraction that produces a desirable line drive current.

38. A data communications system as recited in claim **37** wherein the examining comprises:

determining the fraction at which the noted high and low receiver reference voltages are within specified parameters.

39. A data communications system as recited in claim **37** wherein the examining comprises:

determining the fraction at which the noted high and low receiver reference voltages have a specified relationship with each other.

40. A data communications system as recited in claim **37** wherein the examining comprises:

determining the fraction at which the noted high receiver reference voltage is greater than a first parameter and the noted low receiver reference voltage is less than a second parameter.

41. A data communications system as recited in claim **37** wherein the examining comprises:

determining the fraction at which the difference between the noted high and low receiver reference voltages is greatest.

42. A data communications system as recited in claim **37** wherein the examining comprises:

determining the fraction at which the average of the noted high and low receiver reference voltages is approximately equal to the receiver reference voltage used during normal bus line operation.

43. A data communications system as recited in claim **37**, the control logic performing further actions comprising:

using the desirable line drive current on said one or more data signaling lines.

44. A data communications system as recited in claim **37**, the control logic performing further actions comprising:

finding and storing desirable line drive currents for a plurality of different receiving components.

45. A data communications system as recited in claim **37**, the control logic performing further actions comprising:

finding and storing desirable line drive currents for a plurality of different receiving components; and

using the stored desirable line drive currents with different receiving components.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,546,343 B1
DATED : April 8, 2003
INVENTOR(S) : Batra et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 1,
Line 22, delete "is" after "a".

Column 3,
Line 11, delete "11" before "is".

Signed and Sealed this

Twelfth Day of August, 2003

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US006378056B2

(12) **United States Patent**
Nizar et al.

(10) **Patent No.: US 6,378,056 B2**
(45) **Date of Patent: Apr. 23, 2002**

(54) **METHOD AND APPARATUS FOR CONFIGURING A MEMORY DEVICE AND A MEMORY CHANNEL USING CONFIGURATION SPACE REGISTERS**

(75) Inventors: **Puthiya K. Nizar**, El Dorado Hills; **William A. Stevens**, Folsom, both of CA (US)

(73) Assignee: **Intel Corporation**, Santa Clara, CA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/186,050**

(22) Filed: **Nov. 3, 1998**

(51) **Int. Cl.**[7] **G06F 12/00**
(52) **U.S. Cl.** **711/170**; 711/160; 711/166; 710/10; 710/74; 710/104; 713/1
(58) **Field of Search** 711/160, 166, 711/170; 710/10, 74, 104; 713/1

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,236,207 A * 11/1980 Rado et al. 711/166
5,263,168 A * 11/1993 Toms et al. 713/1
6,003,121 A * 12/1999 Wirt 711/170

OTHER PUBLICATIONS

Rambus Advance Information—Direct RAC Data Sheet dated Aug. 7, 1998.

Rambus Advance Information—Direct Rambus RIMM Module 128 MBytes (64Mx16/18) dated Aug. 20, 1998.

Rambus Advance Information—Direct RMC.d1 Data Sheet dated Aug. 7, 1998.

Rambus Advance Information—Direct RDRAM 64/72-Mbit (256Kx16/18x16d) Datasheet.

Rambus Advance Information—RIMM Serial Presence Detect Application Note dated Oct. 9, 1997.

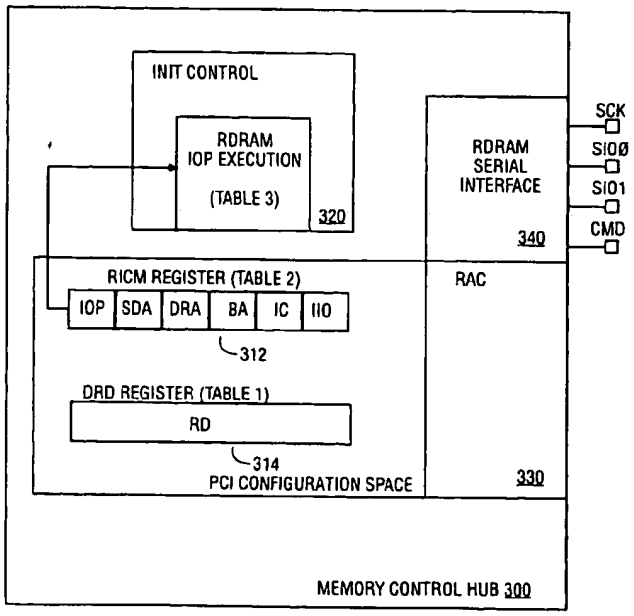
* cited by examiner

Primary Examiner—Do Hyun Yoo
Assistant Examiner—Mehdi Namazi
(74) *Attorney, Agent, or Firm*—Peter Lam

(57) **ABSTRACT**

A method and apparatus for configuring memory devices. A disclosed bus controller includes a storage location and a control circuit. The control circuit is coupled to perform an initialization operation when a value indicating that initialization operation is stored in the storage location. The initialization operation is selected from one of a set of initialization operations that the control circuit is capable of performing.

16 Claims, 12 Drawing Sheets



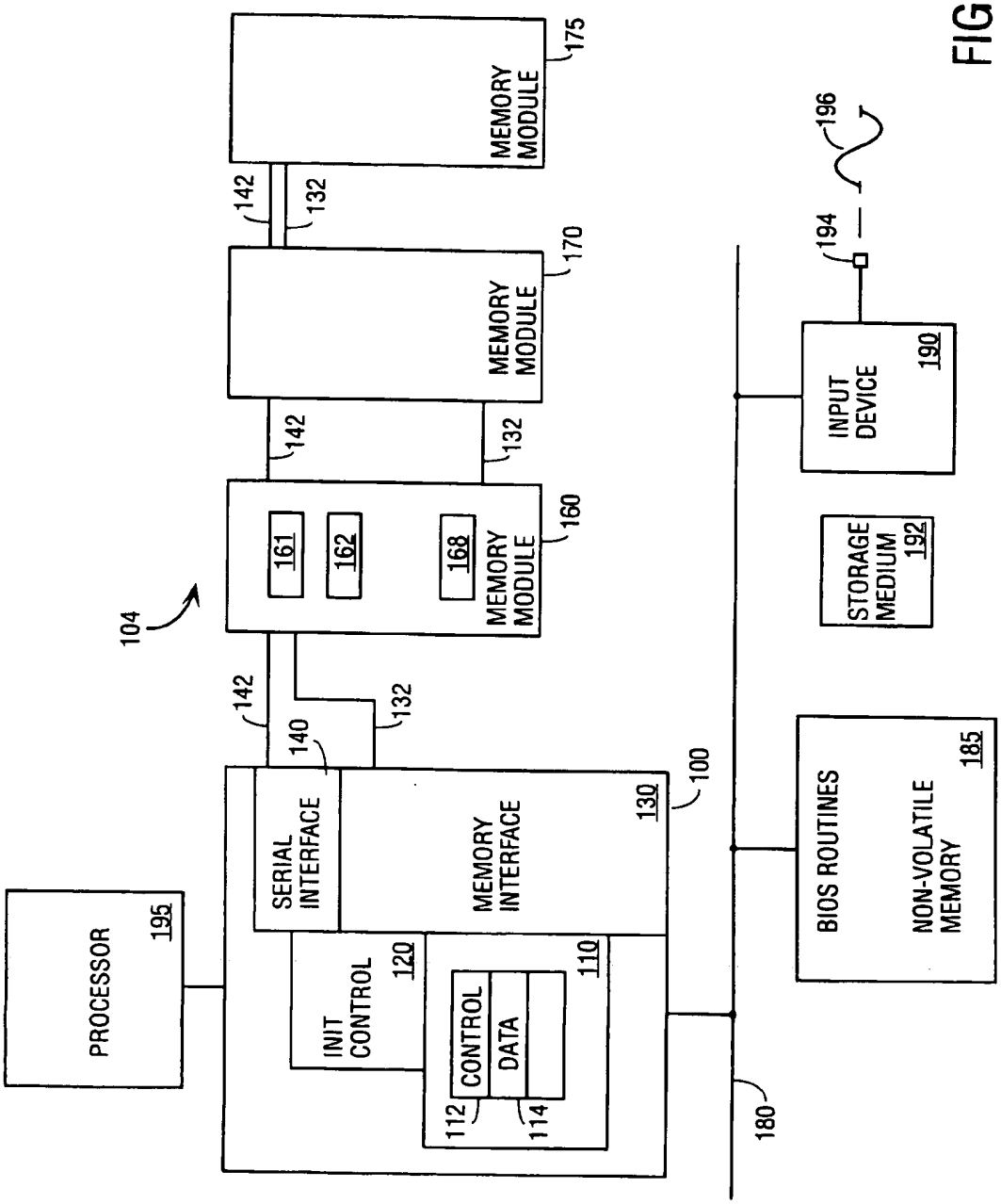

MEMORY MODULE
175
142
132
MEMORY MODULE
170
142
132
161
162
168
MEMORY MODULE
160
104
142
140
132
SERIAL INTERFACE
MEMORY INTERFACE
130
100
PROCESSOR
195
INIT CONTROL
120
CONTROL
DATA
110
112
114
180
196
194
INPUT DEVICE
190
STORAGE MEDIUM
192
BIOS ROUTINES
NON-VOLATILE MEMORY
185

FIG. 1

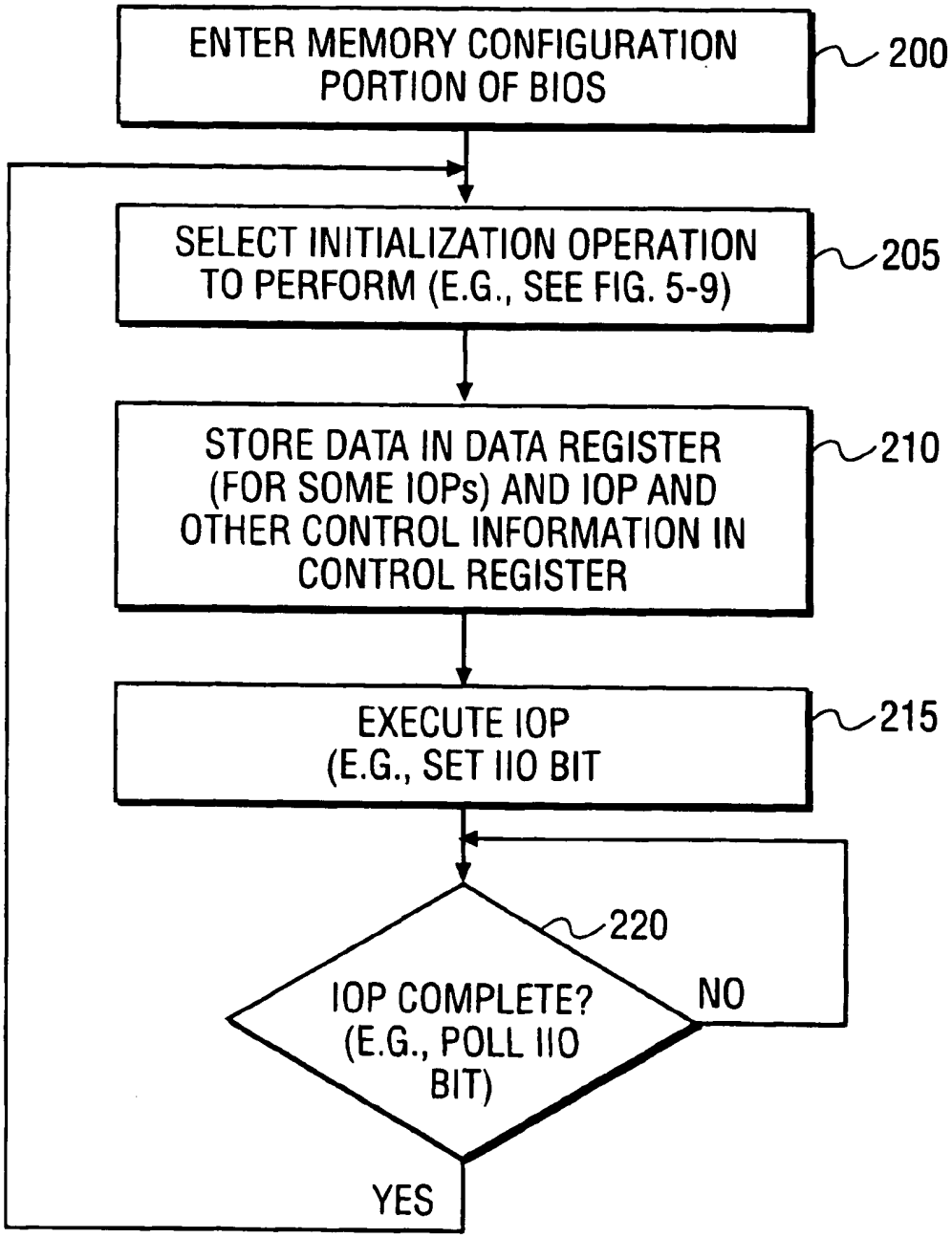

ENTER MEMORY CONFIGURATION PORTION OF BIOS
200
SELECT INITIALIZATION OPERATION TO PERFORM (E.G., SEE FIG. 5-9)
205
STORE DATA IN DATA REGISTER (FOR SOME IOPs) AND IOP AND OTHER CONTROL INFORMATION IN CONTROL REGISTER
210
EXECUTE IOP (E.G., SET IIO BIT
215
220
IOP COMPLETE? (E.G., POLL IIO BIT)
NO
YES

FIG. 2

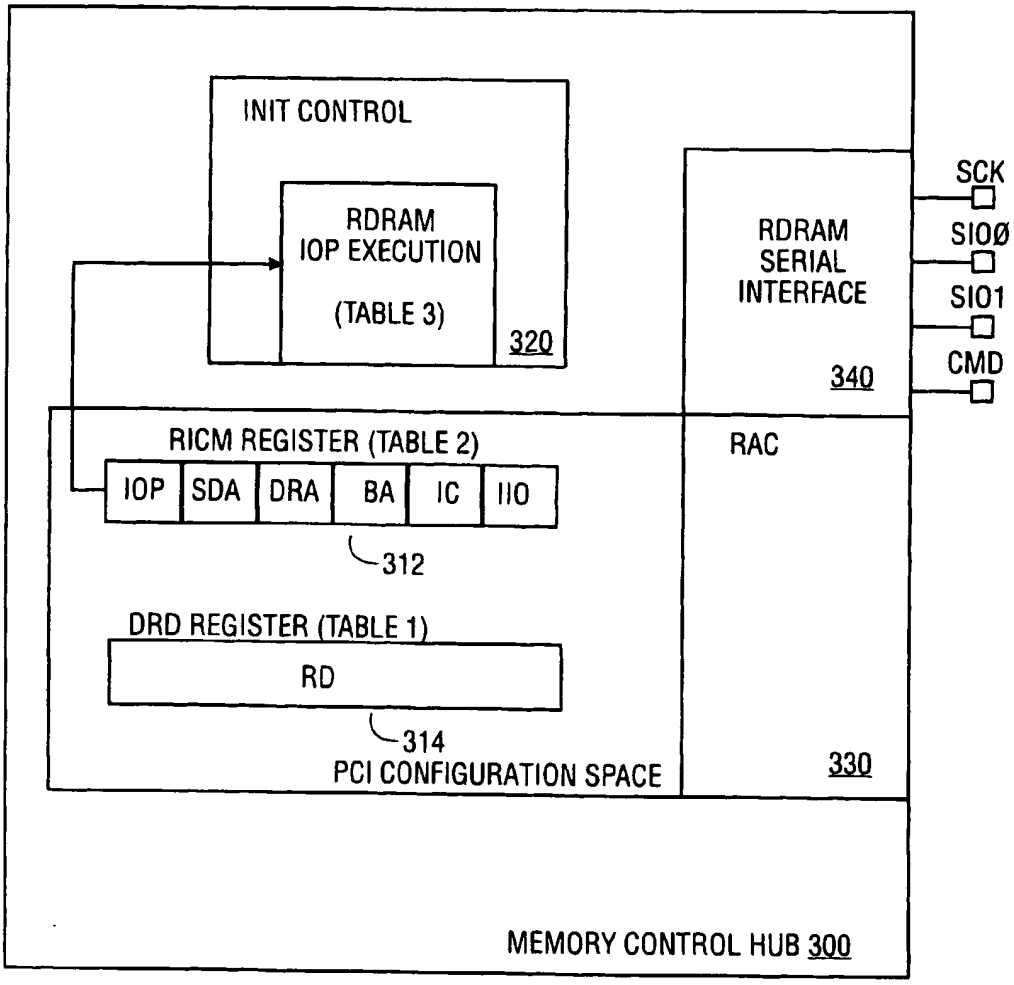

INIT CONTROL
RDRAM
IOP EXECUTION
(TABLE 3)
320
RDRAM
SERIAL
INTERFACE
340
SCK
SIOØ
SIO1
CMD
RICM REGISTER (TABLE 2)
IOP
SDA
DRA
BA
IC
IIO
312
RAC
DRD REGISTER (TABLE 1)
RD
314
PCI CONFIGURATION SPACE
330
MEMORY CONTROL HUB 300

FIG. 3

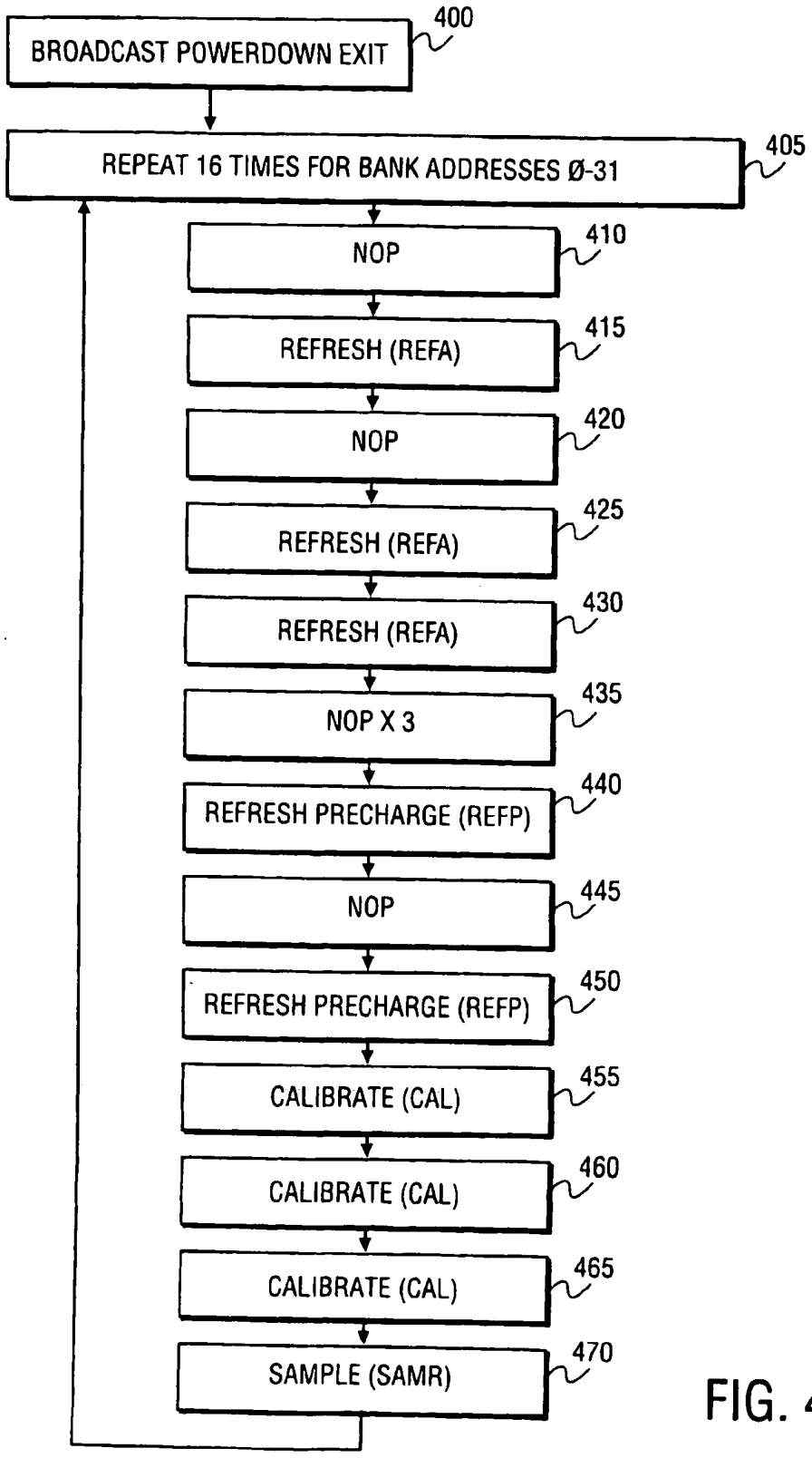

BROADCAST POWERDOWN EXIT
400
REPEAT 16 TIMES FOR BANK ADDRESSES Ø-31
405
NOP
410
REFRESH (REFA)
415
NOP
420
REFRESH (REFA)
425
REFRESH (REFA)
430
NOP X 3
435
REFRESH PRECHARGE (REFP)
440
NOP
445
REFRESH PRECHARGE (REFP)
450
CALIBRATE (CAL)
455
CALIBRATE (CAL)
460
CALIBRATE (CAL)
465
SAMPLE (SAMR)
470

FIG. 4

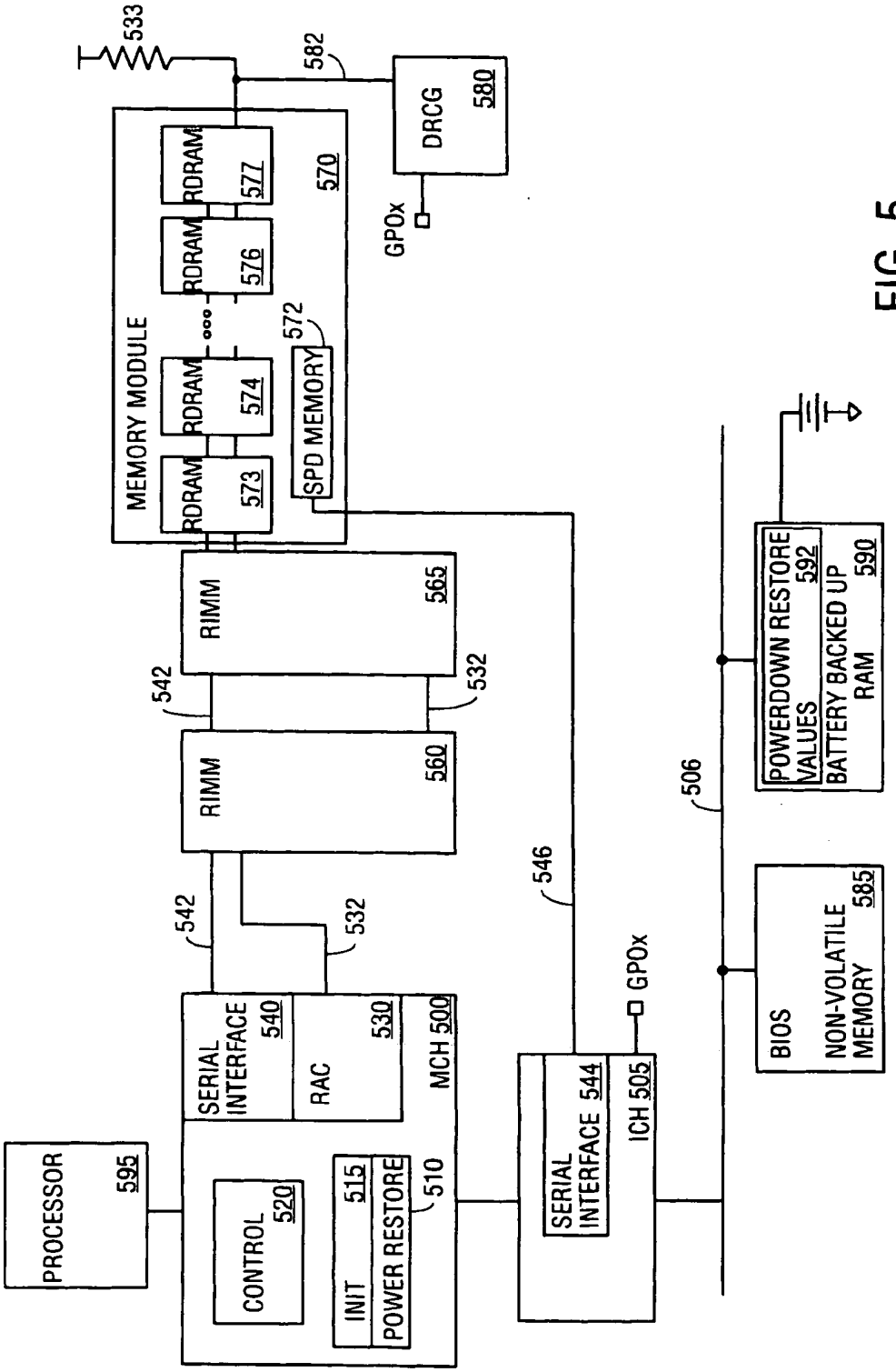

533
582
DRCG
580
GPOx
MEMORY MODULE
570
RDRAM
577
RDRAM
576
RDRAM
574
RDRAM
573
572
SPD MEMORY
RIMM
565
RIMM
560
542
532
542
532
546
506
SERIAL INTERFACE
540
RAC
530
MCH 500
PROCESSOR
595
CONTROL
520
INIT
515
POWER RESTORE
510
SERIAL INTERFACE 544
ICH 505
GPOx
POWERDOWN RESTORE VALUES
592
BATTERY BACKED UP RAM
590
BIOS
NON-VOLATILE MEMORY
585

FIG. 5

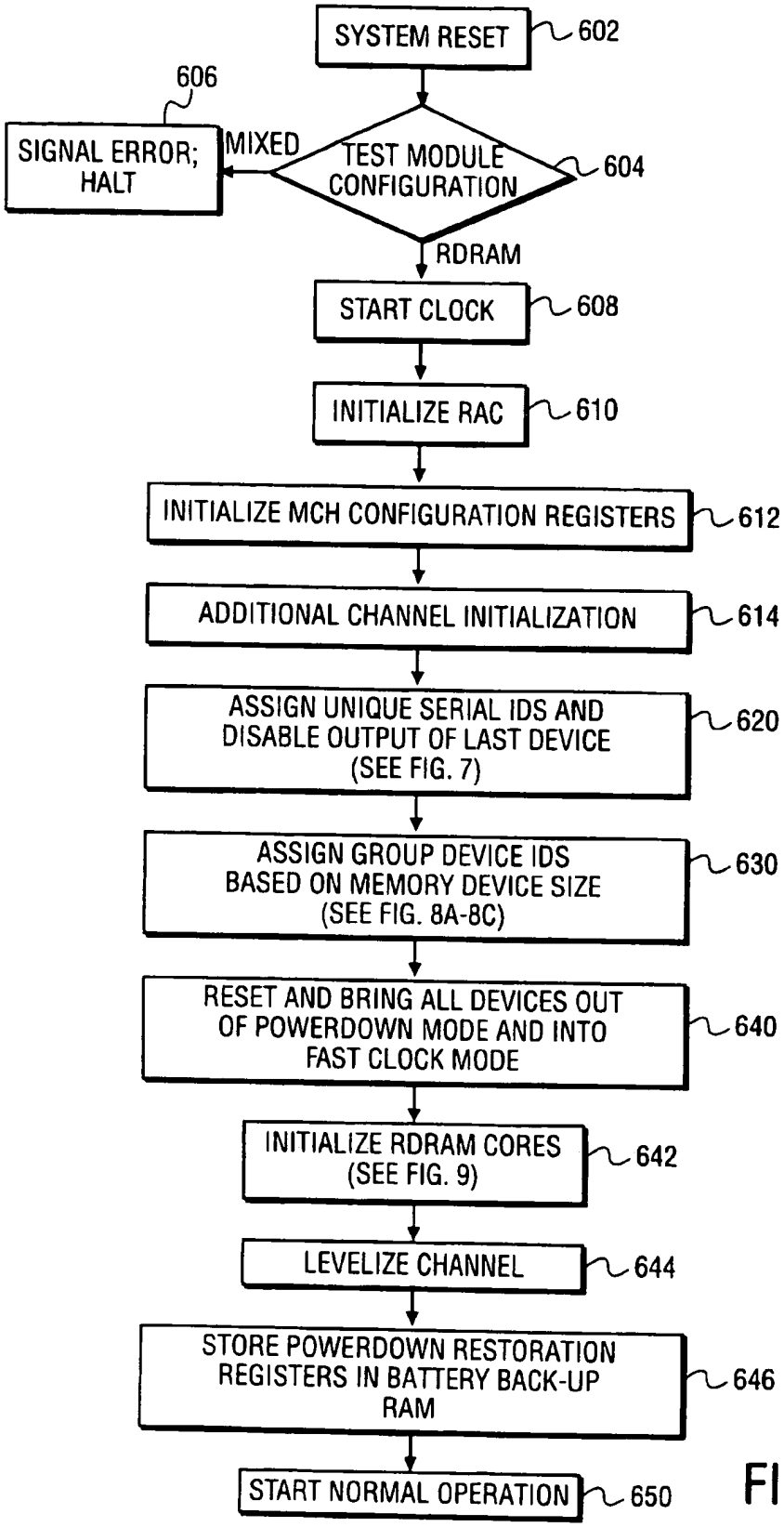

SYSTEM RESET
602
606
SIGNAL ERROR; HALT
MIXED
TEST MODULE CONFIGURATION
604
RDRAM
START CLOCK
608
INITIALIZE RAC
610
INITIALIZE MCH CONFIGURATION REGISTERS
612
ADDITIONAL CHANNEL INITIALIZATION
614
ASSIGN UNIQUE SERIAL IDS AND DISABLE OUTPUT OF LAST DEVICE (SEE FIG. 7)
620
ASSIGN GROUP DEVICE IDS BASED ON MEMORY DEVICE SIZE (SEE FIG. 8A-8C)
630
RESET AND BRING ALL DEVICES OUT OF POWERDOWN MODE AND INTO FAST CLOCK MODE
640
INITIALIZE RDRAM CORES (SEE FIG. 9)
642
LEVELIZE CHANNEL
644
STORE POWERDOWN RESTORATION REGISTERS IN BATTERY BACK-UP RAM
646
START NORMAL OPERATION
650

FIG. 6

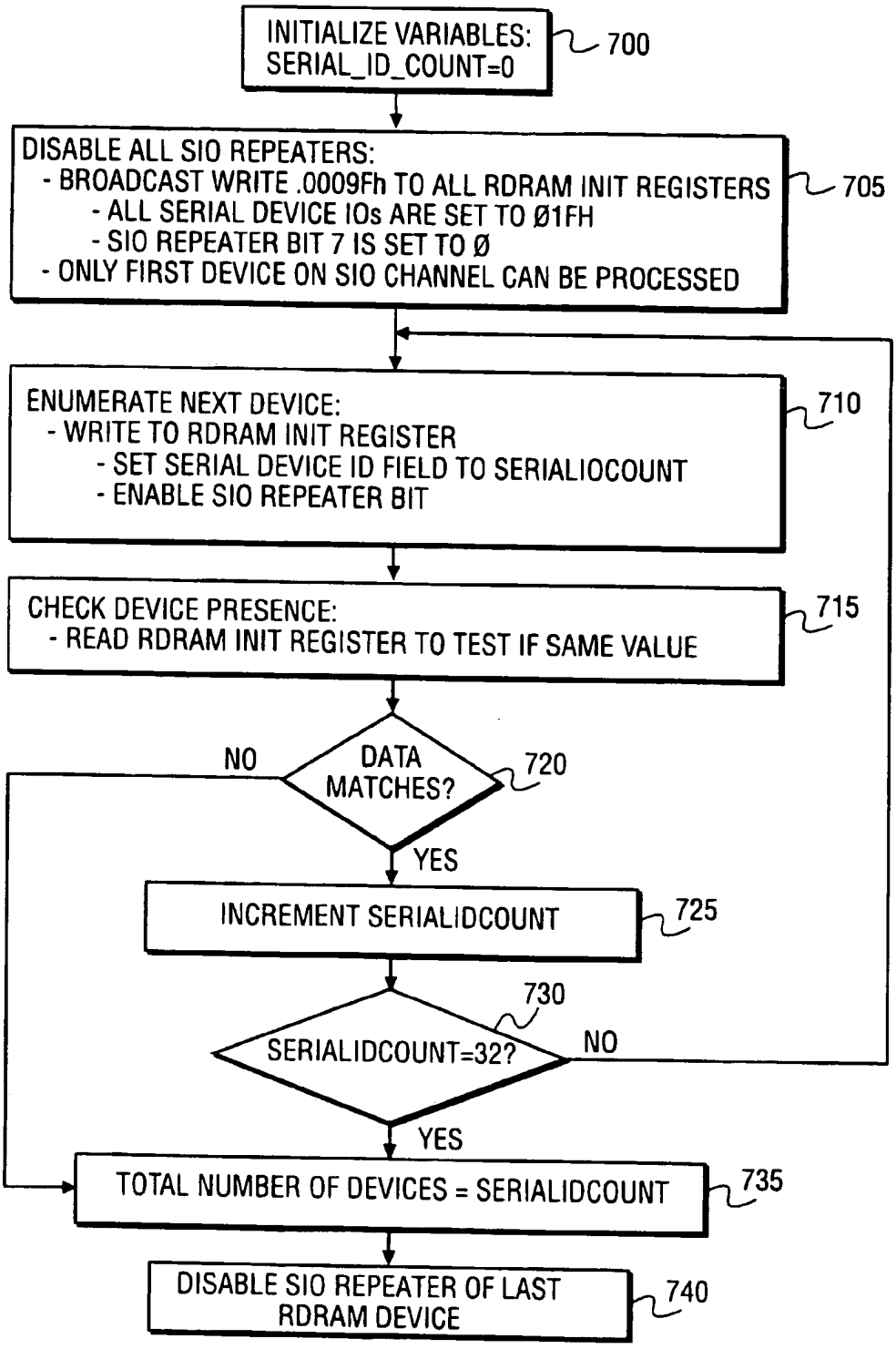

INITIALIZE VARIABLES:
SERIAL_ID_COUNT=0
700
DISABLE ALL SIO REPEATERS:
- BROADCAST WRITE .0009Fh TO ALL RDRAM INIT REGISTERS
- ALL SERIAL DEVICE IOs ARE SET TO Ø1FH
- SIO REPEATER BIT 7 IS SET TO Ø
- ONLY FIRST DEVICE ON SIO CHANNEL CAN BE PROCESSED
705
ENUMERATE NEXT DEVICE:
- WRITE TO RDRAM INIT REGISTER
- SET SERIAL DEVICE ID FIELD TO SERIALIOCOUNT
- ENABLE SIO REPEATER BIT
710
CHECK DEVICE PRESENCE:
- READ RDRAM INIT REGISTER TO TEST IF SAME VALUE
715
DATA MATCHES?
720
NO
YES
INCREMENT SERIALIDCOUNT
725
730
SERIALIDCOUNT=32?
NO
YES
TOTAL NUMBER OF DEVICES = SERIALIDCOUNT
735
DISABLE SIO REPEATER OF LAST RDRAM DEVICE
740

FIG. 7

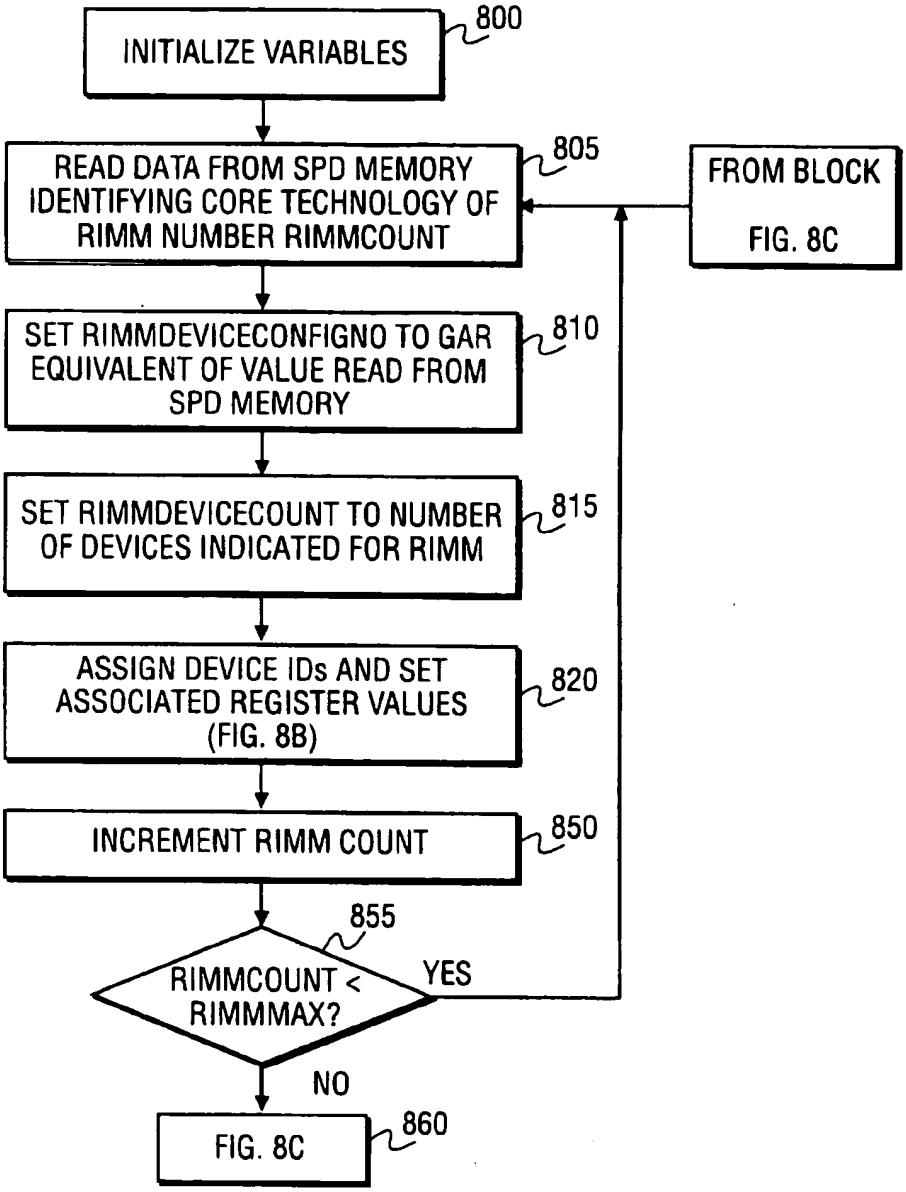

INITIALIZE VARIABLES
800
READ DATA FROM SPD MEMORY IDENTIFYING CORE TECHNOLOGY OF RIMM NUMBER RIMMCOUNT
805
FROM BLOCK
FIG. 8C
SET RIMMDEVICECONFIGNO TO GAR EQUIVALENT OF VALUE READ FROM SPD MEMORY
810
SET RIMMDEVICECOUNT TO NUMBER OF DEVICES INDICATED FOR RIMM
815
ASSIGN DEVICE IDs AND SET ASSOCIATED REGISTER VALUES (FIG. 8B)
820
INCREMENT RIMM COUNT
850
855
RIMMCOUNT < RIMMMAX?
YES
NO
FIG. 8C
860

FIG. 8A

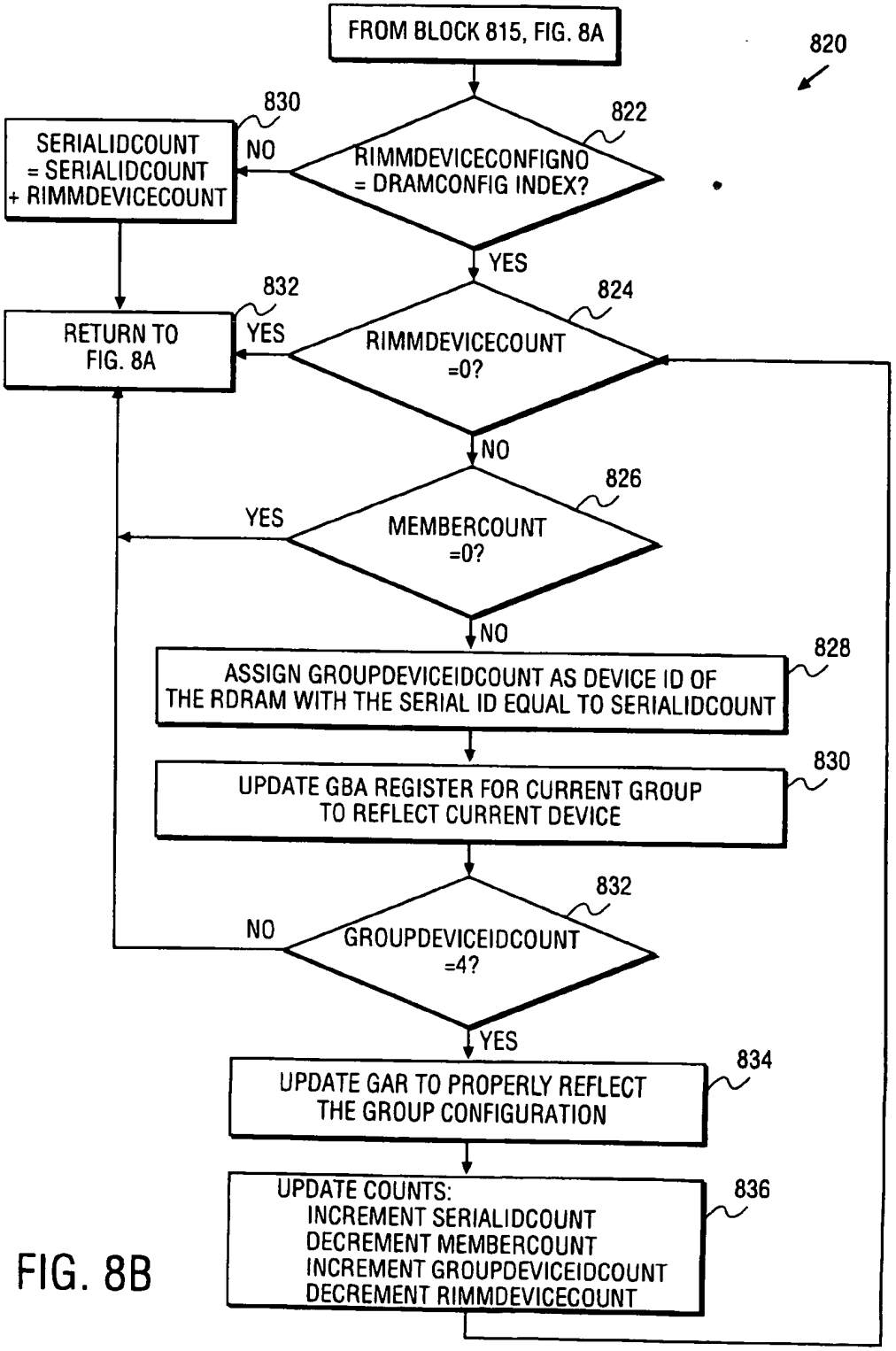

FROM BLOCK 815, FIG. 8A
820
822
RIMMDEVICECONFIGNO = DRAMCONFIG INDEX?
NO
830
SERIALIDCOUNT = SERIALIDCOUNT + RIMMDEVICECOUNT
YES
824
RIMMDEVICECOUNT =0?
YES
832
RETURN TO FIG. 8A
NO
826
MEMBERCOUNT =0?
YES
NO
828
ASSIGN GROUPDEVICEIDCOUNT AS DEVICE ID OF THE RDRAM WITH THE SERIAL ID EQUAL TO SERIALIDCOUNT
830
UPDATE GBA REGISTER FOR CURRENT GROUP TO REFLECT CURRENT DEVICE
832
GROUPDEVICEIDCOUNT =4?
NO
YES
834
UPDATE GAR TO PROPERLY REFLECT THE GROUP CONFIGURATION
836
UPDATE COUNTS:
INCREMENT SERIALIDCOUNT
DECREMENT MEMBERCOUNT
INCREMENT GROUPDEVICEIDCOUNT
DECREMENT RIMMDEVICECOUNT

FIG. 8B

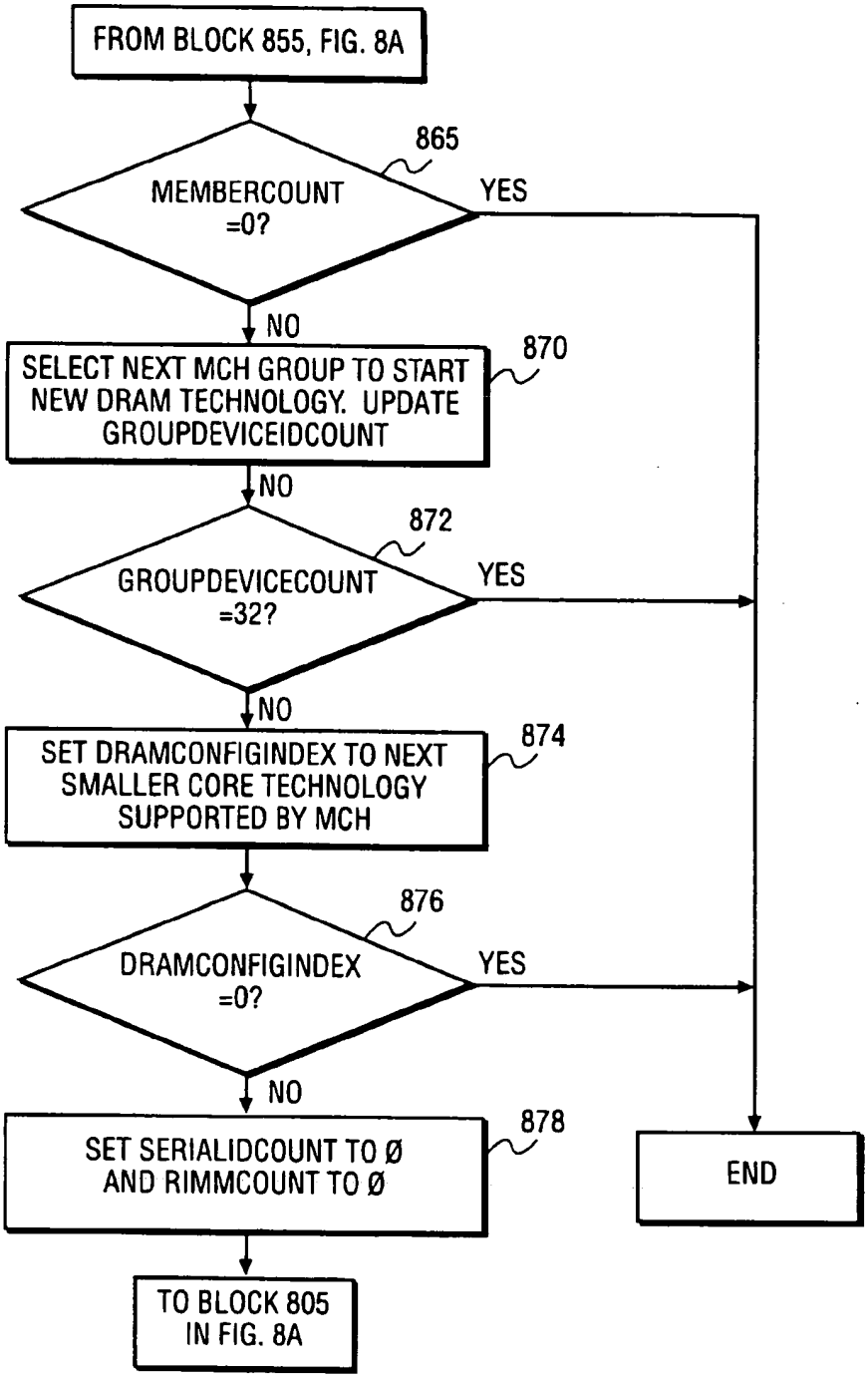

FROM BLOCK 855, FIG. 8A
865
MEMBERCOUNT =0?
YES
NO
870
SELECT NEXT MCH GROUP TO START NEW DRAM TECHNOLOGY. UPDATE GROUPDEVICEIDCOUNT
NO
872
GROUPDEVICECOUNT =32?
YES
NO
874
SET DRAMCONFIGINDEX TO NEXT SMALLER CORE TECHNOLOGY SUPPORTED BY MCH
876
DRAMCONFIGINDEX =0?
YES
NO
878
SET SERIALIDCOUNT TO Ø AND RIMMCOUNT TO Ø
END
TO BLOCK 805 IN FIG. 8A

FIG. 8C

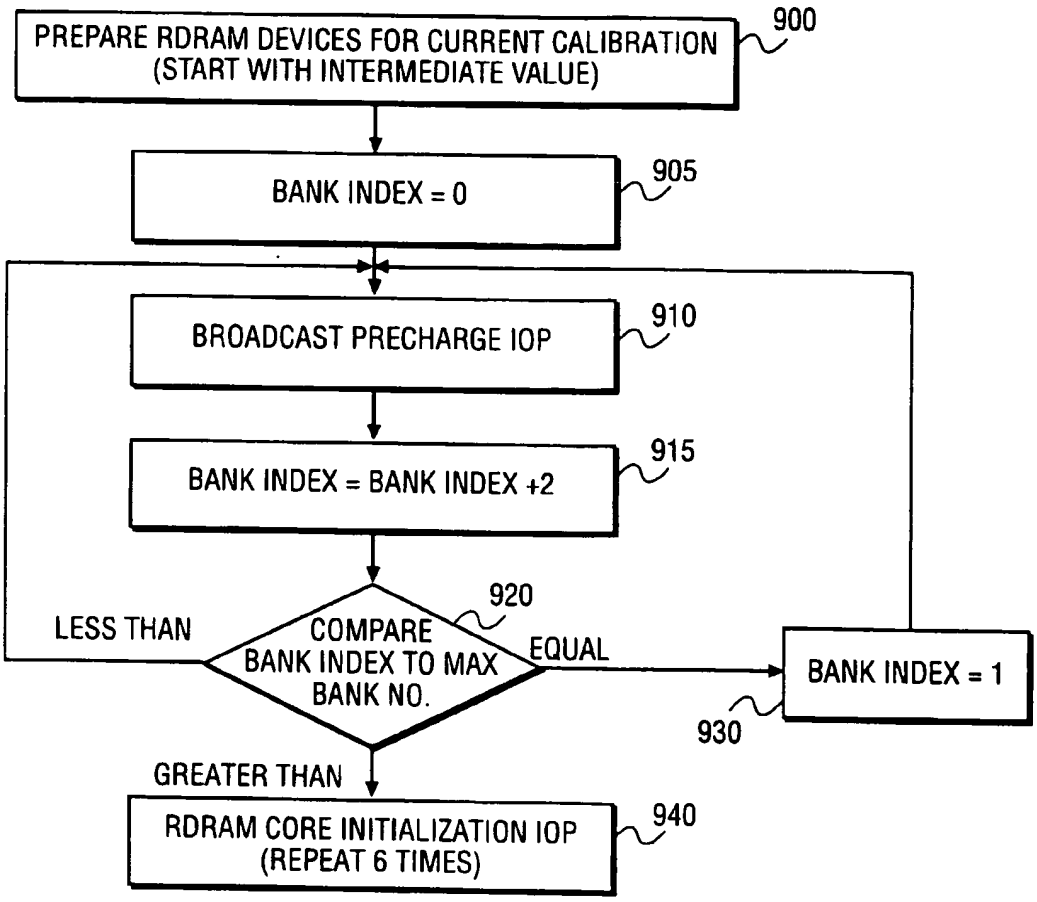

PREPARE RDRAM DEVICES FOR CURRENT CALIBRATION (START WITH INTERMEDIATE VALUE)
900
BANK INDEX = 0
905
BROADCAST PRECHARGE IOP
910
BANK INDEX = BANK INDEX +2
915
COMPARE BANK INDEX TO MAX BANK NO.
920
LESS THAN
EQUAL
BANK INDEX = 1
930
GREATER THAN
RDRAM CORE INITIALIZATION IOP (REPEAT 6 TIMES)
940

FIG. 9

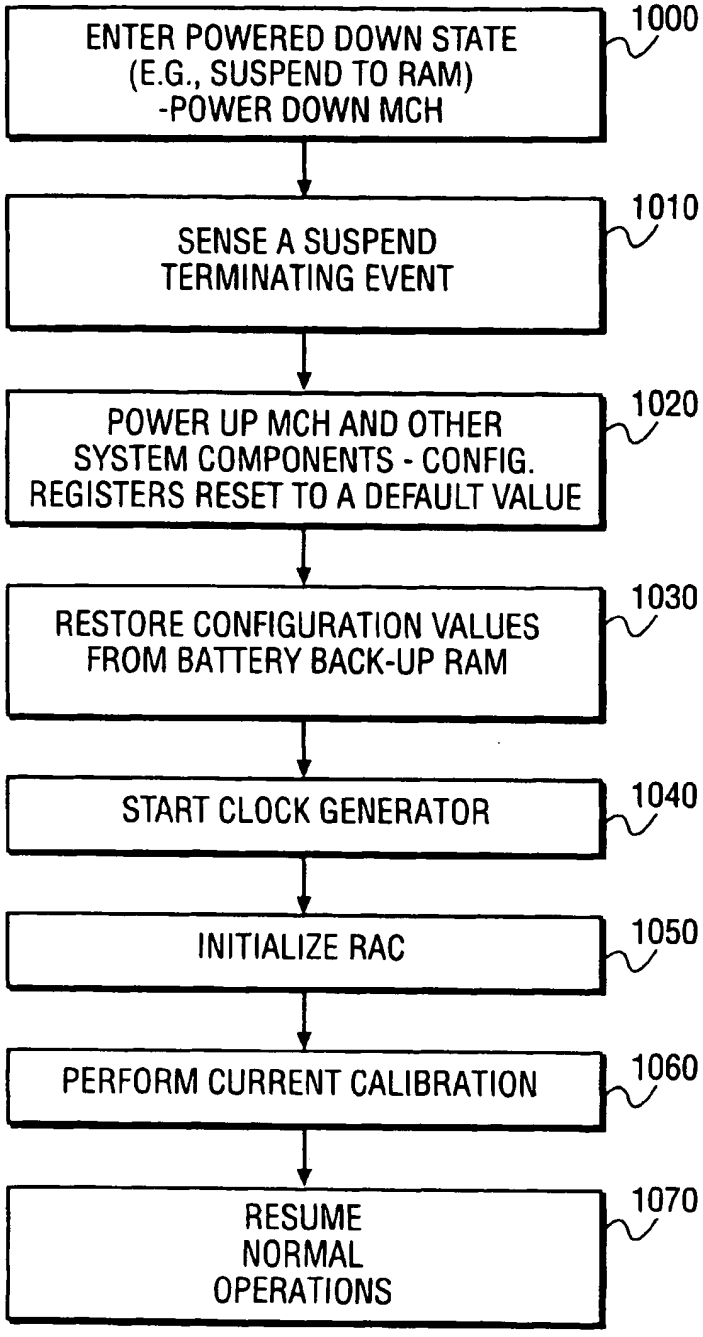

ENTER POWERED DOWN STATE (E.G., SUSPEND TO RAM) -POWER DOWN MCH
1000
SENSE A SUSPEND TERMINATING EVENT
1010
POWER UP MCH AND OTHER SYSTEM COMPONENTS - CONFIG. REGISTERS RESET TO A DEFAULT VALUE
1020
RESTORE CONFIGURATION VALUES FROM BATTERY BACK-UP RAM
1030
START CLOCK GENERATOR
1040
INITIALIZE RAC
1050
PERFORM CURRENT CALIBRATION
1060
RESUME NORMAL OPERATIONS
1070

FIG. 10

METHOD AND APPARATUS FOR CONFIGURING A MEMORY DEVICE AND A MEMORY CHANNEL USING CONFIGURATION SPACE REGISTERS

RELATED APPLICATIONS

This application is related to an application Ser. No. 09/186,046, entitled "A Method And Apparatus For Levelizing Transfer Delays For A Channel Of Devices Such As Memory Devices In A Memory Subsystem," application Ser. No. 09/186,051, entitled "A Method And Apparatus For Configuring And Initializing A Memory Device And A Memory Channel," and application Ser. No. 09/186,049, entitled "A Method And Apparatus For Restoring A Memory Device Channel When Exiting A Low Power State," all of which are filed concurrently herewith.

BACKGROUND

1. Field of the Invention

The present disclosure pertains to the field of data processing systems. More particularly, the present disclosure pertains to initializing or configuring a memory devices in a memory channel.

2. Description of Related Art

Memory devices and memory subsystems typically have certain initialization steps and/or register values that need to be programmed prior to normal operation. Recommended steps and values are often detailed in a memory specification provided to system designers who design other system hardware interfacing with the memory devices. Unfortunately, memory specifications may change, and new initialization sequences and/or register values may be desirable.

To implement a new system compliant with a revised specification, expensive hardware changes may be required. Additionally, if more optimal initialization values or sequences are later discovered, they may not be implemented due to the high overhead of making hardware changes. Present systems do not provide a flexible memory initialization technique that may advantageously allow initialization sequences to be altered or optimized after hardware components have been completed.

One channel (i.e., a bus configuration) which requires a significant amount of initialization prior to proper operation is a Rambus™ Direct Rambus Dynamic Random Access Memory Channel (a Direct RDRAM™ Channel). This channel is described in detail in documentation available from Rambus Corporation of Mountain View, Calif. RDRAM memories and memory controllers interfacing with a Rambus channel have various registers that need to be programmed during the initialization process.

One method of initializing a direct RDRAM channel is described in the RMC.d1 Data Sheet, which describes a Direct Rambus™ Memory Controller (RMC). The RMC, however, utilizes an "Init Block" as a part of an application specific integrated circuit (ASIC) to implement initialization procedures (see FIG. 7, p. 16 and p. 72, et seq.). The Init Block is simply activated using two init signals and thereafter performs a routine hard-wired into the Init Block ASIC hardware.

Thus, the prior art in general may not provide an adequate method and apparatus for configuring a set of memory devices. A hard-wired solution such as the RMC ASIC inherently lacks the flexibility to allow rapid alteration of initialization sequences in order to optimize initialization or

to contend with a changing memory specification. A flexible memory initialization technique could advantageously be utilized in a Direct Rambus™ bus or in any other memory system requiring significant initialization or configuration.

SUMMARY

A method and apparatus for configuring memory devices is disclosed. A disclosed bus controller includes a storage location and a control circuit. The control circuit is coupled to perform an initialization operation when a value indicating that initialization operation is stored in the storage location. The initialization operation is selected from one of a set of initialization operations that the control circuit is capable of performing.

BRIEF DESCRIPTION OF THE FIGURES

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings.

FIG. 1 illustrates one embodiment of a system using configuration registers in a memory controller to designate initialization operations for memory initialization.

FIG. 2 illustrates a flow diagram of programming and executing initialization operations in one embodiment of the system of FIG. 1.

FIG. 3 illustrates one embodiment of a memory control hub that performs memory initialization according to values loaded into control and data registers.

FIG. 4 illustrates a flow diagram for a memory device core initialization operation.

FIG. 5 illustrates one embodiment of a system implementing an initialization flow shown in FIGS. 6–9.

FIG. 6 illustrates a flow diagram of one embodiment of an overall initialization sequence for the memory subsystem of the system shown in FIG. 5.

FIG. 7 illustrates one embodiment of a serial device identification process.

FIG. 8A illustrates one embodiment of a first portion of a group device identification process.

FIG. 8B illustrates one embodiment of a group device assignment process (e.g., block 820 in FIG. 8A).

FIG. 8C illustrates one embodiment of a second portion of the group identification process from FIG. 8A.

FIG. 9 illustrates one embodiment of a memory device core initialization process.

FIG. 10 illustrates one embodiment of the process of returning from a suspend-to-RAM power management state.

DETAILED DESCRIPTION

The following description provides a method and apparatus for initializing a memory device and a memory channel. In the following description, numerous specific details such as register names, memory types, bus protocols, specific types of components, and logic partitioning and integration choices are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. In other instances, control structures and gate level circuits have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement the necessary logic circuits without undue experimentation.

Using the presently disclosed techniques, efficient and flexible memory initialization may be performed. Control

and data registers may be programmed, thereby causing a memory control hub (MCH) to perform initialization operations (IOPs) according the values loaded in the registers. Since the registers may be programmed by software such as a basic input/output system (BIOS), the initialization may be altered with relative ease.

FIG. **1** illustrates one embodiment of a system utilizing registers to perform memory initialization. The system includes a processor **195** and a memory subsystem **104** that are coupled to a memory control hub (MCH) **100**. Also coupled to the MCH **100** is a secondary bus **180** having coupled thereto an input device **190** and a non-volatile memory **185** containing BIOS routines. In some embodiments, either or both of the non-volatile memory **185** and the input device **190** may be coupled to the MCH **100** by a second control hub (not shown).

In the illustrated embodiment, the memory subsystem **104** includes three memory modules **160**, **170**, and **175** coupled to the MCH **100** via a serial bus **142** and a memory bus **132** (also referred to as a channel). Each memory module may contain a set of individual memory devices. For example, the memory module **160** includes at least memory devices **160**, **161**, and **168**. In one embodiment, the memory devices **160**, **161**, and **168** are Rambus DRAMs (RDRAMs), the memory modules are Rambus In-line Memory Modules (RIMMs), and the channel operates according to protocols defined for RIMMs and RDRAMs.

The MCH **100** includes a control register **112** and a data register **114** which may be used for initialization purposes. An initialization control circuit **120** executes initialization operands (IOPs) which are programmed into the control register **112**. The control register **112** typically includes other fields to specify information about initialization operations, and some of the operations specified by the IOPs involve data exchange with devices in the memory subsystem (e.g., writing and reading of memory device control registers or otherwise generating control signals).

A serial interface circuit **140** generates serial command and data sequences on the serial bus **142**. Some of the commands executed by the initialization control circuit **120** send commands and/or data to the memory subsystem via the serial bus **142**. Control registers, including device registers for identification numbers, may be read and written via the serial interface circuit **140**.

A memory interface circuit **130** translates memory data to and from data packets which are exchanged with the memory subsystem. In one embodiment, the memory interface circuit is a Rambus ASIC Cell (RAC) functioning substantially as described in the "Direct RAC Data Sheet" available from Rambus Corporation of Mountain View, Calif. Briefly, the RAC converts the Rambus Signal Level (RSL) signals on the channel (bus **132**) to signals which can be processed by other portions of the MCH **100**. Similarly, the RAC converts the memory controller signals to RSL signal which can be processed by memory devices on the Rambus channel.

A sequence of initialization events for the system of FIG. **1** is illustrated in FIG. **2**. As the system is reset or turned on, the BIOS typically performs various initialization operations. In block **200**, the BIOS reaches the memory configuration portion. Depending on the type of memory and the intended mode of usage, initialization operations will be selected (block **205**) by the BIOS in a particular sequence. More details of one embodiment of an initialization sequence for a system utilizing RDRAMs are discussed with respect to FIGS. **5–9**.

As indicated in block **210**, data (if any) for the particular initialization operation is stored in the data register **114**, and the initialization operand itself with other control information is stored in the control register **112**. In some embodiments, the BIOS may perform this function by writing to peripheral component interconnect (PCI) configuration registers. Alternatively, other registers may be used, or general purpose memory locations either within or without the MCH may be the control register. In fact, the control register may be any storage location accessible to the MCH prior to memory initialization that is capable of storing sufficient bits for IOPs and any other needed control information.

The initialization operation may commence automatically when the proper initialization operation and/or control information are programmed into the control register **112**. For example, the execution of the initialization operation indicated in block **215** may be accomplished by setting an initiate initialization operation (IIO) bit when the initialization operand is loaded into the control register **112**. The IIO bit may be a field of the control register **112** so the same register write transaction may set the IIO bit and provide the IOP.

Completion of the initialization operation may be signaled in any manner sufficient to alert or inform the BIOS that the initialization operation is complete. For example, the MCH may automatically clear the IIO bit when the initialization operation completes. If the BIOS polls the IIO bit, it may determine when the initialization operation completes as indicated in block **220**. If the initialization operation has not completed, the BIOS may continue polling the IIO bit. If the initialization operation has completed, the BIOS may select the next initialization operation in the initialization sequence in block **205**.

The input device **190** may either accept program instructions from a computer storage device **192** (e.g., an optical or magnetic disk or other storage device) or from a network or communications interface **194**. BIOS code (i.e., computer instructions) causing the system to implement the disclosed techniques may be programmed into the non-volatile memory **185** in several ways. The BIOS may be programmed when the system is manufactured or may be later delivered via a computer readable medium through the input device **190**.

In cases where the BIOS is later delivered, the instructions may be delivered via a computer readable medium. With an appropriate interface device **190**, either an electronic signal or a tangible carrier is a computer readable medium. For example, the computer storage device **192** is a computer readable medium in one embodiment. A carrier wave **196** carrying the computer instruction is a computer readable medium in another embodiment. The carrier wave **196** may be modulated or otherwise manipulated to contain instructions that can be decoded by the input device **190** using known or otherwise available communication techniques. In either case, the computer instructions may be delivered via a computer readable medium.

FIG. **3** illustrates additional details of a memory controller hub (MCH) **300**. Details of specific register names, locations, sizes, field definitions, and initialization operations are given for one embodiment below. Other embodiments will be apparent to those of skill in the art. Several of the operations below invoke specific commands defined by Rambus in the 64/72-Mbit Data Sheet and the Direct RAC data sheet. These defined operations are operations that the Rambus RAC itself sends to RDRAMs when appropriate

control signals are sent to the RAC. As detailed below, this embodiment of the MCH **300** invokes known RAC commands by previously unavailable hardware and in new methods or sequences.

In this exemplary embodiment, the MCH **300** includes a RAC **330** and a serial interface **340**. The serial interface **340** uses clock (SCK), serial frame (CMD), and bi-directional serial I/O pins (SIO0 and SIO1) to read and write RDRAM configuration registers as well as to perform other IOPs. The MCH also includes a device register data (DRD) register **314**. The DRD register **314** is at address offset 90–91h in PCI configuration space, the default value is 0000h (16 bits), and the register is a read/write register. The fields of the DRD register are shown in Table 1.

TABLE 1

An Embodiment of the DRD Register

Bit	Description
15:0	Register Data (RD): Bits 15:0 contain the 16 bit data to be written to a RDRAM register or the data read from a RDRAM register as a result of IOP execution. Data will be valid when the IIO bit of RICM register transitions from 1 to 0

The MCH **300** also includes a RDRAM Initialization Control Management (RICM) Register **312**. The RICM Register is at address offset 94–96h in PCI configuration space, the default value is 000000h (24 bits), and the register Is a read/write register. The fields of the RICM register for this embodiment are set forth in Table 2.

TABLE 2

An Embodiment of the RICM Register

Bit	Description
23	Initiate Initialization Operation (IIO): When set to 1, the execution of the initialization operation specified by the IOP field (below) starts. After the execution is completed, the MCH clears the IIO bit to 0. A software program should check to see if this bit is 0 before writing to it. Data from operations which specify a register data read from the RDRAM will be valid in the DRD register when the IIO bit is cleared to 0.
22:21	RESERVED: These bits are not used for normal initialization operations.
20	Initialization Complete (IC): BIOS sets this bit to 1 after initialization of the RDRAM memory array is complete.
19	Broadcast Address (BA): When BA is set to 1, the initialization operation (IOP) is broadcast to all devices in the channel. When BA is set to 1, the SDA field (below) is not used.
17:9	Device Register Address (DRA): This field specifies the register address for the register read and write operations.
8:4	Serial Device/Channel Address (SDA): This 5 bits field specifies the following: • the serial device ID of the RDRAM device for RDRAM Register Read, RDRAM Register Write, RDRAM Set Reset, RDRAM Clear Reset and RDRAM Set Fast Clock Mode IOP commands. • the device ID for Powerdown Entry, Powerdown Exit, Nap Entry, Nap Exit, Current Calibrate and Current Calibrate & Sample IOP commands. • the bank address for Refresh and Precharge IOP commands
18, 3:0	Initialization Opcode (IOP): This field specifies the initialization operation to be done on a RDRAM device or the MCH RAC. Bits[18, 3:0] Operation Specified 0 0 0 0 0 RDRAM Register Read 0 0 0 0 1 RDRAM Register Write 0 0 0 1 0 RDRAM Set Reset 0 0 0 1 1 RDRAM Clear Reset 0 0 1 0 0 RDRAM Set Fast Clock Mode 0 0 1 0 1 Reserved 0 0 1 1 0 RDRAM Temperature Calibrate Enable and then Temperature Calibrate 0 0 1 1 1 to 0 1 1 1 1 Reserved 1 0 0 0 0 RDRAM Core Initialization (RCI) 1 0 0 0 1 RDRAM SIO Reset 1 0 0 1 0 RDRAM Powerdown Exit 1 0 0 1 1 RDRAM Powerdown Entry 1 0 1 0 0 RDRAM "Current Cal" and "Current Cal + Sample"

TABLE 2-continued

An Embodiment of the RICM Register

Bit	Description	
	1 0 1 0 1	Manual Current Calibration of MCH RAC
	1 0 1 1 0	Load MCH RAC control register with data from DRD register
	1 0 1 1 1	Initialize MCH RAC
	1 1 0 0 0	RDRAM Nap Entry
	1 1 0 0 1	RDRAM Nap Exit
	1 1 0 1 0	RDRAM Refresh
	1 1 0 1 1	RDRAM Precharge
	All other combinations are reserved.	
	More details on the operations specified by IOP field is shown in Table 3 below.	

Also illustrated in FIG. **3** is an initialization control circuit **320** which includes an RDRAM IOP execution circuit **325**. Details of the various IOPs executed by the control circuit **320** are illustrated in Table 3. In Table 3, the broadcast address (BA) field (bit **19**) and the SDA field (bits **8**:**4**) are listed either as one of the following:

TABLE 3

IOP Operation Details

Bits [18, 3:0]	Operation Name	BA	SDA	Details
00000	RDRAM Register Read	0	x	This IOP performs the serial read of the RDRAM register specified by SDA and DRA fields. The data read will be available in DRD register when the IIO bit is cleared to 0.
00001	RDRAM Register Write	x	x	This IOP performs the serial write of the RDRAM register specified by SDA and DRA fields. The write data is provided in the DRD register. A write operation to all RDRAM devices on the channel can be performed by setting the BA field to 1.
00010	RDRAM Set Reset	x	x	This IOP performs the serial setting of a reset bit in an RDRAM device specified by the SDA field. The setting of the reset bit begins a period for the RDRAM device to reset and prepare to respond to all other defined operations. The Set Reset IOP also puts the RDRAM in active mode.
00011	RDRAM Clear Reset	x	x	This IOP performs the serial clearing of the reset bit in an RDRAM device specified by SDA field. The Clear Reset operation puts the device into Powerdown state. A minimum of 4 SCK cycles must pass after the SIO Request Packet before the RDRAM device is allowed to exit this Powerdown state. The Clear Reset operation may not be issued before greater than 16 SCK cycles have occurred after the Set Reset operation.
00100	RDRAM Set Fast Clock Mode	x	x	The Set Fast Clock Mode operation prepares the RDRAM device to transmit and receive data on RSL signals using RDRAM clock (RCLK).
00110	RDRAM Temperature Calibrate Enable and then Temperature Calibrate	1	x	Upon receiving this IOP, the MCH issues a "Temperature Calibrate Enable" SIO request packet followed immediately by a "Temperature Calibrate" SIO Request packet to all RDRAMs.
10000	RDRAM Core Initialization (see FIG. 4)	NE	NE	Upon receiving this IOP command the MCH does the following: 1. Broadcast Powerdown Exit. 2. Initialize all RDRAM cores of all RDRAM devices on the channel. 3. Broadcast Temp Cal Enable and Temp Cal. 4. Broadcast NAP entry (if bit 6 (PBS) of DRAMC register is 1). 5. If IC bit (bit 20) of RICM register is set to 1 along with this command, then the MCH enables RDRAM Refresh, RDRAM Current Cal, RDRAM Temp Cal, and RDRAM DLL Refresh logic after this command completes.
10001	RDRAM SIO Reset	NE	NE	This IOP sends an SIO pin initialization sequence to all RDRAMs. When this operation occurs the SIO0 pin on the RDRAM is configured as input and SIO1 pin is configured as output. Additionally, the SIO repeater bit

TABLE 3-continued

IOP Operation Details				
Bits [18, 3:0]	Operation Name	BA	SDA	Details
				is set to 1.
10010	RDRAM Powerdown Exit	x	x	Upon receiving this IOP, the MCH initiates a Powerdown exit sequence for the RDRAM device specified by SDA and BA fields. The SDA field should contain the device ID, not the serial device ID.
10011	RDRAM Powerdown Entry	x	x	Upon receiving this IOP, the MCH sends a Powerdown Entry PCP packet to the RDRAM device specified by SDA and BA fields. The SDA field should contain the device ID, not the serial device ID.
10100	RDRAM "Current Cal" and "Current Cal + Sample"	x	x	Upon receiving this IOP, the MCH sends three Current Calibrate SCP packets followed by one Current Calibrate and Sample SCP packet to the RDRAM device specified by SDA field.
10101	Manual Current Calibration of MCH RAC	NE	NE	Upon receiving this IOP, the MCH initiates a manual Current calibration operation of MCH RAC.
10110	Load MCH RAC control register with data from DRD register	NE	NE	Upon receiving this IOP, the MCH loads the MCH RAC control register with the data from the DRD register.
10111	Initialize MCH RAC	NE	NE	Upon receiving this IOP, the MCH initializes the MCH RAC. The MCH RAC initialization includes Power Up sequence, Current Calibration and Temperature Calibration of the MCH RAC. After executing this command, the MCH enables the periodic Current and Temperature Calibration of the MCH RAC even if the IC bit is not set to 1.
11000	RDRAM Nap Entry	x	x	Upon receiving this IOP, the MCH sends a Nap Entry PCP packet to the RDRAM device specified by SDA and BA fields. The SDA field should contain the device ID, not the serial device ID.
11001	RDRAM Nap Exit	x	x	Upon receiving this IOP, the MCH initiates a Nap exit sequence for the RDRAM device specified by SDA and BA fields. The SDA field should contain the device ID, not the serial device ID.
11010	RDRAM Refresh	1	x	Upon receiving this IOP, the MCH sends a Refresh PCP packet to the specified bank of all RDRAM devices. The bank address is specified by SDA field.
11011	RDRAM Precharge	1	x	Upon receiving this IOP, the MCH sends a Precharge PCP packet to the specified bank of all RDRAM devices. The bank address is specified by SDA field.

NE: This field has no effect on the initialization operation
0: This field is to be set to 0 for this initialization operation.
1: This field is to be set to 1 for this initialization operation.
x: This field should be programmed as appropiate for the particular initialization operation.

Details of operations conducted by one embodiment of the initialization control circuit **320** in response to receiving the RDRAM Core Initialization IOP (10000b) are shown in FIG. **4**. In block **400**, a broadcast powerdown exit command is issued on the bus. Next, as per block **405**, the sequence indicated by blocks **410** to **470** is repeated sixteen times for bank addresses zero to thirty-one. These numbers may be appropriate for a memory subsystem having one hundred and twenty-eight current calibration levels and up to thirty-two banks. In other embodiments, a different number of repetitions may be used if, for example, a larger or smaller number of current calibration levels are available. Similarly, differing numbers of banks may be available in different systems.

In block **410**, no operation is performed to ensure that the powerdown exit is complete and that the refresh operation (REFA command) is properly performed in block **415**. In block **420**, another no operation command is executed, followed by two more refresh operations (REFA) in blocks **425** and **430**. Three more no operation commands are executed in block **435**, allowing sufficient time to pass before a refresh precharge (REFP) command occurs. After another no operation command in block **445**, another refresh precharge (REFP) command is executed in block **450**.

A calibrate (CAL) command is next executed in block **455**. This command calibrates (drives) I_{OL} current for the presently indicated device. As indicated in blocks **460** and **465**, this operation may be repeated twice. Then, as indicated in block **470**, a sample (SAMR) command is executed. The sample command updates the I_{OL} current for the presently indicated device. Until all sixteen repetitions for the thirty-two banks are performed, this process is repeated.

Initialization Sequence

With the above initialization operations, a system may be initialized. For example, the system shown in FIG. **5**, which implements a Rambus Direct RDRAM channel, may be initialized. In this system, a memory controller **500** (also referred to as a memory control hub or MCH) orchestrates initialization activities. The memory controller also initiates specific ROW/COLUMN packets on the channel. A serial interface **540** may be used to communicate with devices on the channel. For example, clock (SCK), serial frame (CMD), and bi-directional serial I/O (SIO0 and SIO1) pins may be used to read and write RDRAM configuration registers as well as to perform other IOPs.

The memory controller includes a Rambus ASIC Cell (RAC) **530**, a control circuit **520**, and a variety of registers. The registers include a initialization registers **515**, which are used to initialize the system memory, and powerdown restoration registers **510**. The powerdown restoration registers contain timing and other information crucial to operating the memory channel. In other words, the powerdown restoration registers are simply registers that need to be restored after powering down the memory controller **500** in order to resume accesses to the memory channel. The registers may be PCI configuration registers.

The memory channel includes RIMM modules **560**, **565**, and **570** that are connected to the MCH **500** by a control and data bus **532** and a serial bus **542**. The control and data bus **532** may be terminated by a resistive termination **533**, and a Direct Rambus Clock Generator (DRCG) **580** may be provided at the far end of the channel from the MCH **500** to provide clock signals over signal lines **582**.

Additionally, the system includes an Input/Output control hub (ICH) **505** which couples the MCH to a secondary bus **506** and may contain a second serial interface circuit **544** for interfacing with a second serial bus **546**. A serial presence detect (SPD) memory **572** (a non-volatile memory such as an electrically erasable and programmable read only memory) for each module may be read via the serial interface **544** according to a serial presence detect protocol. The SPD memory **572** may provide information such as timing information., device organization, and device technology about each particular memory module. More details of the SPD protocol are discussed in the "Serial Presence Detect Application Brief" as well as the Direct Rambus™ RIMM™ Module and the 64/72 Mbit Direct RDRAM™ data sheets available from Rambus.

In one embodiment, the serial bus **546** is an I2C bus such as a system management bus (SMBus). This embodiment includes clock (SMBCLK) and data (SMBDATA) signals that follow the industry defined System Management Bus (SMBus) protocol as defined in the System Management Bus Specification, Revision 1.0, available from the Smart Battery Implementer's Forum.

The ICH has general purpose outputs (GPOs) which are used to control various system functions such as setting the frequency of the DRCG **580**. A non-volatile memory **585** containing the BIOS may be coupled to the secondary bus **506**, as well as a battery backed-up random access memory **590**. The battery backed-up memory **590** may store powerdown restoration configuration values **592** for the MCH powerdown registers **510** so the MCH can resume accessing the RDRAM channel without performing the full initialization sequence detailed below.

Briefly, the initialization process may be summarized as follows. After power up reset, the configuration information from Serial Presence Detection (SPD) data on the RIMMs in a channel is read. For example, a storage device, SPD memory **572**, stores configuration information for the RDRAMs **573**, **574**, **576**, and **577** on the RIMM **570**. The memory controller configuration registers are programmed with the appropriate values from the SPD information, and then the RDRAM device IDs are programmed such that each RDRAM device can be uniquely identified and accessed by the memory controller. Once a device has been initialized, it can be used.

Each RDRAM device has two identification numbers that are used to uniquely select a device on the channel, the Serial Device ID, and the Group Device ID. These two IDs are used for distinct operations on the RDRAM channel. The serial device ID is used to select devices when the memory controller is sending initialization operations on the SCK, SIO, and CMD signals of the RDRAM channel. The group device ID is used by the memory controller to select a device when sending ROW packets and COLUMN packets on RQ[7:0] signals of the RDRAM channel. Both the serial device ID and the group device ID are programmed after reset and before devices may be individually addressed by initialization operations (IOPs) and ROW/COLUMN packets, respectively.

Looking at the initialization process of the Rambus channel in more detail, a particular sequence may be followed to achieve correct operation of the RDRAM devices on the channel. FIG. **6** illustrates a flow diagram for proper channel initialization in one embodiment, and Table 4 enumerates some of the variables used in this initialization flow.

TABLE 4

	Variables Used in Initialization	
Name	Width (bits)	Description
RIMMMax	2	Maximum number of RIMMs present.

TABLE 4-continued

Variables Used in Initialization

Name	Width (bits)	Description
		0 No RIMMs present 1–3 1–3 RIMM(s) present
RIMMCount	2	Counter used during initialization to select a RIMM.
RIMMDeviceCount	5	Number of RDRAM devices in a particular RIMM.
MemberMax	5	Maximum number of devices present on a channel 0–31 1–32 RDRAM devices present on the channel
MemberCount	5	Counter used during group device ID enumeration to indicate # of devices that have been assigned group IDs.
SerialIDCount	5	Serial Device ID index used to select devices on a channel. 0–31 Maps to serial device ID 0–31
GroupDeviceIDCount	5	Group Device ID index used during group device ID enumeration to assign a Group Device ID to the next RDRAM device. 0–31 Maps to group device ID 0–31
RIMMDeviceConfigNo	8	Byte indicating RDRAM technology definition. Bit definition matches GAR register.
DRAMConfigIndex	3	Index into table of DRAM technologies supported by MCH. Used during group device ID enumeration assign group IDs to RDRAMs in a technology descending order.
MchTrdly	3	Temporary storage of maximum Mch Trdly during channel levelization procedure. Bit definition matches the MCH's tRDLY field in the MCH RDT register.
DeviceTestAddress	32	32-bit CPU address used to test a RDRAM device during channel levelization.
TempIndex	8	Temporary index used during algorithm.

In block **602**, system reset occurs. The MCH resets all its state machines and prepares for initialization. In block **604**, memory module configuration of the system is verified. The BIOS reads SPD data to determine the memory configuration. If only RIMMs are present, the RDRAM initialization sequence may proceed with block **608**. If mixed memory modules are present, an error is posted to the user and the system is halted as indicated in **606**.

The clock generator is started in block **608**. This operation may be accomplished by software querying the SPD data of every RIMM module present on the motherboard and determining a channel frequency at which all RIMMs may operate. The DRCG **580** may be set to the proper frequency by a general purpose output (i.e., GPOx as shown in FIG. **5**) from the ICH **505**. In one embodiment, the BIOS waits at least 8 ms between this step and the MCH RAC initialization.

As indicated in block **610**, the MCH RAC is next initialized. The channel clock from the DRCG should be stable prior to MCH RAC initialization. The MCH RAC initialization is accomplished by executing the MCH RAC initialization IOP. The RAC initialization IOP performs basic initialization to prepare the internal RAC of the memory controller for normal operation.

In one embodiment, the BIOS provides a time out of 5 ms for the IIO bit to clear after the MCH RAC initialization IOP. If the IIO bit is not cleared by the MCH after 5 ms, the BIOS should report the error, and the channel is unusable. An additional 5 ms delay may be added after the MCH clears the IIO bit due to completion of the MCH RAC initialization IOP. This allows sufficient time for the MCH clocks to stabilize and lock. Also in some embodiments, a bus in the RAC may need to be cleared before other operations commence. This may be accomplished by executing the MCH RAC Control Register Load IOP (DRD=00000h). It may also be possible to perform the RAC initialization at a later point in the initialization sequence in some embodiments.

As indicated in block **612**, a number of MCH configuration registers may next be initialized. In one embodiment, the paging policy register RMC idle timer (PGPOL RIT) field (MCH 052h [2:0]) is set to 001b to ensure no pages are closed during channel levelization (discussed below). The PGPOL RIT field sets the number of host bus clocks that the memory controller will remain in the idle state before all open pages are closed, and a value of zero indicates that there will be an infinite latency before the memory controller starts closing pages.

Additionally, in some embodiments, operating pools may be used to group RDRAMs based on defined RDRAM states. In order to reduce operating power, the RDRAM devices may be grouped into two operating pools called "Pool A" and "Pool B." In one embodiment, up to eight devices may be in Pool A at a time. In this embodiment, up to four out of eight devices in Pool A may be in Active Read/Write or Active states at a time, and the devices in Pool A are in either Active Read/Write, Active, or Standby states.

The maximum number of devices in Pool A is programmable and is specified by a PAC field of the RDRAM power management register (RPMR) register (MCH 053h). All devices that are not in Pool A are members of Pool B. All devices in Pool B are either in the Standby or Nap state. The state of the devices in Pool B is specified by a PBS field of a DRAM control (DRAMC) register (MCH 051h). In one embodiment, the RPMR register is set to 00h, selecting a pool A of 1 device only, and Pool B operation is set for standby operation (MCH 051h [6]=0).

Next, as indicated in block **614**, additional channel initialization may be performed. This may include performing an SIO (serial interface) reset using the SIO reset IOP, and allowing sufficient delay for completion of the SIO reset

sequence. Additionally, other registers which may need to be initialized for proper operation may be set at this point. For example, in some embodiments, a Test77 register may need to be written to with a zero value after the SIO reset as specified on page 37 of the Direct RDRAM 64/72 Mbit Data Sheet (execute a Broadcast SIO Register Write IOP: TEST77, DRA=4 Dh, DRD=0000h).

Serial Device ID Assignment

As indicated in block **620**, serial device identification values (IDs) may be assigned next. In general, the software uniquely identifies each device on the channel to allow initialization operations to be targeted at individual devices. The serial device ID for each RDRAM is stored in the RDRAM INIT register (index 21h) in bits **4-0**. After SIO reset, the default value of the serial device ID is 1 Fh in all RDRAMs on the channel. Also, after reset, the Serial Repeater (SRP bit (RDRAM 021h [7]) is set to 1, enabling each RDRAM to propagate SIO data received on SIO**0** to the RDRAM's SIO**1** pin, passing the SIO packet to the next RDRAM device. Since all devices have the same serial device ID after reset, an individual device may not be accessed prior to assigning unique serial IDs.

Further details of the serial device enumeration performed by one embodiment are shown in FIG. **7**. In block **700**, the variable SerialIDCount is initialized to zero. Next, as indicated in block **705**, the SIO repeaters of all devices on the channel are disabled (Broadcast SIO Register Write IOP. INIT, DRA=21h, DRD=001Fh). This operation causes all serial device IDs to be set to 01fh. The SIO repeater bit is set to zero, so only t he first device on the SIO channel can be accessed.

Starting with block **710**, the process loops through all devices on the channel and assigns a unique ID to each. The serial ID of the current device is set to SerialIDCount and the SIO repeater bit is enabled (SIO Register Write IOP: INIT, SDCA=1Fh, DRA=21h, DRD=0080h+SerialIDCount). Next, whether the device is actually present and functioning in the system is tested as indicated in block **715**. The RDRAM INIT register is read to determine if the same value which was just written is properly read back out (SIO Register Read IOP. INIT, SDCA=SerialIDCount, DRA=21h).

If the data matches (as tested in block **720**), serialIDcount is incremented (block **725**), and the serialIDCount is checked to see whether a maximum number of devices (e.g., thirty-two) have been given IDs (block **730**). If the serialIDCount still indicates a valid serial ID, the next device is identified in block **705**.

If the serialIDCount exceeds the maximum permissible value, or if the data did not match in block **720**, then the last device has been given an ID, and a variable tracking the total number of devices may be set to the serialIDCount as indicated in block **735**. Finally, to disable any additional devices beyond the last permitted device, the SIO repeater of the RDRAM with the highest serial ID is disabled as shown by block **740**. Accordingly, any additional devices (i.e., improperly functioning devices or devices beyond the maximum, e.g., thirty-two) do not receive commands and therefore should not respond. As an additional check, the SPD information on the RIMMs may be examined to determine if the final device count is correct.

Group Device ID Assignment

Returning to FIG. **6**, after the unique serial IDs have been assigned and the SIO output of the last device disabled, group IDs are assigned based on memory device size as indicated in block **630**. In one embodiment, the MCH supports up to thirty-two RDRAM devices and eight groups. Each group has up to four devices and has a group boundary access register (GBA) to define the group ID and the upper and lower addresses for each group. Thus, each GBA register may be programmed with a group ID and a nine bit upper address limit value. Unpopulated groups may have a value equal to the previous group and a group size of zero.

Additionally, the flowchart in FIGS. **8A–8C** illustrates one embodiment of the process of enumerating group device IDs indicated in block **630**. As indicated in block **800** in FIG. **8A**, a number of variables are initialized. Variables SerialIDCount, GroupDeviceIDCount, RIMMCount, RIMMDeviceCount, and RIMMDeviceConfigNo are initialized to zero. A DRAMConfigIndex variable is initialized to a value indicating the largest core technology supported by the MCH.

As indicated in block **805**, data is read from the SPD memory of a module (module number RIMMCount) identifying the core technology of that module. This information may include the number of rows per device, the number of columns per device, the number of banks per device, and whether the banks are dependent or independent. Next, as indicated in block **810**, the RIMMDeviceConfigNo is set by translating the core technology value read from the SPD into a value in a Group Architecture (GAR) register equivalent value.

Next, as indicated in block **815**, the RIMMDeviceCount variable is set to the number of devices indicated by the SPD memory for that RIMM. Thereafter, the device IDs may be assigned and associated register values set as indicated in block **820**. Further details of the process indicated in block **820** for one embodiment are shown in FIG. **8B**.

In general, the enumeration process adds the number of RDRAM devices on a RIMM to the first Serial ID and then counts down until the RIMM is finished. Therefore, as indicated in block **822**, whether RIMMDeviceConfigNo equals the DRAMConfigIndex is tested to determine whether group device IDs have been assigned for all devices in a particular core technology. If they are unequal, all devices for that core technology have group IDs, so the SerialIDCount is set to SerialIDCount plus RIMMDeviceCount (as indicated in block **830**) and the process returns to FIG. **8A** as indicated in block **832**. Additionally, if RIMMDeviceCount is zero (as tested in block **824**) or MemberCount is zero (as tested in block **826**), there are no more devices to give group IDs and the process returns to FIG. **8A** as indicated in block **832**.

If RIMMDeviceCount and MemberCount are not zero, a GroupDeviceIDCount is assigned to be the group device ID of the RDRAM with the serial ID equal to the present value of SerialIDCount as indicated in block **828**. Next, the current group boundary address register (GBA) is updated to reflect the addition of the new device to this group as indicated in block **830**. This may be accomplished by adding a value indicative of the device size to the previous value stored in that GBA register.

Next, the GroupDeviceIDCount is compared to four (the maximum number of devices per group in one embodiment) in block **832**. If the group is full, the MCH Group Architecture Register (GAR) for that group is updated as indicated in block **834**. The GAR is updated to properly indicate the group configuration (i.e., the number of banks and the DRAM technology (size)). In block **836**, SerialDeviceIDCount is incremented, MemberCount is decremented,

GroupDeviceIDCount is incremented, and RIMMDeviceCount is decremented. The process then returns to block **824**.

Returning to FIG. 8A, if either RIMMDeviceCount or MemberCount is zero, RIMMCount is incremented as indicated in block **850**. If RIMMCount is less than a maximum RIMMCount, as tested in block **855**, then the process returns to block **805**. If the RIMMCount has reached the last RIMM, the process continues in FIG. 8C as indicated by block **860**.

Turning to FIG. 8C, if MemberCount is zero (as tested in block **865**), the device ID enumeration process ends. If, however, MemberCount is not zero, the next MCH group is selected to start enumerating the devices in the next DRAM technology as indicated in block **870**. GroupDeviceIDCount may be updated by adding three and performing a logical AND operation of the resulting value and 0FFFCh.

If GroupDeviceIDCount is a maximum number devices allowed in the channel (e.g., thirty-two as tested in block **872**), then the group ID enumeration process ends. If, however, fewer devices have been given group ID numbers, the DRAMConfigIndex is set to the next smallest core technology supported by the MCH as indicated in block **874**. If the DRAMConfigIndex indicates that there are no smaller core technologies supported (e.g., DRAMConfigIndex is zero as tested in block **876**), then the ID enumeration process ends. If there are more core technologies, serialIDCount and RIMMCount are reset to zero, as indicated in block **878**, and the process returns to block **805** in FIG. 8A.

The psuedo-code below indicates operations that may be used to perform the group ID enumeration indicated by block **630** of FIG. **6** in one embodiment.

630. Enumerate MCH device groups.

630.1. Loop through RIMM SPD memory and group the devices on the RIMMs. The largest technology devices must be grouped in the lowest groups, with the technology size decreasing as the group #s increase.

630.1.1. Set MemberCount=MemberMax

630.1.2. Set SerialIDCount=0. This is the Serial Device ID counter

630.1.3. Set GroupDeviceIDCount=0. This is the Group Device ID counter

630.1.4. Set RIMMCount=0. This is the RIMM counter

630.1.5. Set RIMMDeviceCount=0. This is the counter for the # of devices on a RIMM.

630.1.6. DRAMConfigIndex=Largest technology supported by MCH

630.1.7. Compute RIMM #RIMMCount's core technology

630.1.7.1. RIMMDeviceConfigNo=core technology read from RIMMs SPD.

630.1.8. RIMMDeviceCount=# of RDRAM devices in RIMM #RIMMCount, read from the RIMM's SPD EEPROM.

630.1.9. Assign group device IDs and program MCH GAR and GBA registers for RIMM.

630.1.630.1. If RIMMDeviceConfigNo!=DRAMConfigIndex, break to **630.1.9.1**

630.1.9.2. If RIMMDeviceCount=0, break to **630.1.10**

630.1.9.3. If MemberCount=0, break to **630.1.10**

630.1.9.4. SIO Register Write IOP. DEVID, SDCA=SerialIDCount, DRA=40h, DRD=GroupDeviceIDCount.

630.1.9.5. Program MCH GBA [GroupDeviceIDCount SHR 2]=MCH GBA [GroupDeviceIDCount SHR 2-1]+RIMM #RIMMCount device size.

630.1.9.6. If GroupDeviceIDCount AND 011b=0

630.1.9.6.1. Program MCH GAR [GroupDeviceIDCount SHR 2]=RIMMDeviceConfigNo

630.1.9.7. Increment GroupDeviceIDCount

630.1.9.8. Increment SerialIDCount

630.1.9.9. Decrement MemberCount

630.1.9.10. Decrement RIMMDeviceCount

630.1.9.11. Go to step **630.1.9.2**

630.1.10. Increment RIMMCount

630.1.11. If RIMMCount<RIMMMax, go to step **630.1.7**

630.1.12. If MemberCount=0 then break to step **10**

630.1.13. Select next group for next RDRAM technology.

630.1.13.1. GroupDeviceIDCount=(GroupDeviceIDCount+011b) AND 011b

630.1.14. If GroupDeviceIDCount=32 then break to step **10**

630.1.15. DRAMConfigIndex=next smallest DRAM technology

630.1.16. If DRAMConfigIndex=0, then break to step **10**

630.1.17. SerialIDCount=0

630.1.18. RIMMCount=0

630.1.19. Go to step **630.1.7**. This will begin searching the RIMMs for the next smallest RDRAM technology.

Returning to FIG. **6**, after the group IDs have been assigned, the individual RDRAM devices may be brought out of powerdown mode and put into fast clock mode for normal operation as indicated in step **640**. The individual RDRAM timing registers in the MCH and RDRAMs may be programmed. The REFB and REFR RDRAM control registers may also be initialized (Broadcast SIO Register Write IOP. REFB, DRA=41h, DRD=0000h; Broadcast SIO Register Write IOP. REFR, DRA=42h, DRD=0000h).

The RDRAM devices may be reset by executing a Broadcast Set Reset IOP, followed by an appropriate delay (e.g., 32 us), then executing a Clear Reset IOP, also followed by an appropriate delay (e.g., 4 us) to allow for the reset operation to complete. The RDRAMs are brought out of powerdown by executing a broadcast RDRAM power down exit IOP, and the fast clock mode is entered by executing a broadcast RDRAM Set Fast Clock Mode Initialization IOP.

Thereafter, the RDRAM cores may be initialized as indicated in block **642**. Further details of one embodiment of the RDRAM core initialization are shown in FIG. **9**. As indicated in block **900**, the RDRAM devices are prepared for current calibration by writing an intermediate value to the appropriate RDRAM registers (Broadcast SIO Register Write IOP. CCA, DRA=43h, DRD=0040h; Broadcast SIO Register Write IOP. CCB, DRA=44h, DRD=0040h). Forty hexadecimal may be an appropriate intermediate value in an embodiment that has one hundred and twenty-seven possible current calibration levels. Starting at this intermediate value limits the total number of calibration cycles needed since the calibration value could only be off by approximately half than the full range of calibration values.

Next, precharge operations are performed on each bank of each RDRAM device. To perform the precharge operations, the MCH counts up through the banks by two, first precharging odd banks, and then even ones. A bank index is set to zero in block **905**. A broadcast precharge IOP is then executed as indicated in block **910**. The bank index value is incremented by two as indicated in block **915**, and the broadcast precharge is repeated for even banks until the bank

index is found to be equal to a maximum number of banks (e.g., thirty two) in block **920**.

Once the maximum number of banks is reached, the bank index is set to one (as indicated in block **930**), and all odd banks are precharged. Once the bank index exceeds the maximum number of banks, the RDRAM Core Initialization IOP is executed six times as indicated in block **940**.

Channel Levelization

Returning to FIG. **6**, after the initialization of the RDRAM cores in block **642**, the channel may be levelized as indicated in block **644**. This process involves equalizing the sum of the RDRAM read response time and a propagation delay from the RDRAM to the MCH for all RDRAMs. In other words, once the channel is levelized, all RDRAMs will provide data at the memory controller in the same number of bus cycles.

The following psuedo-code indicates a sequence of steps that may be performed in one embodiment to implement the levelization process indicated in block **644**.

644. Levelize the Rambus channel
- 644.1. Phase 1: Determine MCH tRDLY field value.
  - 644.1.1. SerialIDCount=MemberMax
  - 644.1.2. MchTrdly=0
  - 644.1.3. Program MCH RDT:TRDLY field=MchTrdly.
  - 644.1.4. Compute the 32 bit address to test the RDRAM device for levelization.
    - 644.1.4.1. SIO Register Read IOP. DEVID, SDCA=SerialIDCount, DRA=40h
    - 644.1.4.2. The DRD (MCH 090h [15:0]) now contains the RDRAM's Device ID
    - 644.1.4.3. DeviceTestAddress=MCH GBA[(DRD SHR 2)−1] SHL 23+((DRD AND 011b) * device size in bytes (from GAR[DRD SHR 2])
  - 644.1.5. Do QWORD write operation to address DeviceTestAddress with TestPattern.
  - 644.1.6. Do QWORD read operation to address DeviceTestAddress
  - 644.1.7. If data read!=TestPattern
    - 644.1.7.1. Increment MCH RDT:TRDLY field.
    - 644.1.7.2. If MCH RDT:TRDLY field<=4 then break to step **644.1.5**.
  - 644.1.8. Else (if data read=TestPattern)
    - 644.1.8.1. MchTrdly data read from MCH RDT:TRDLY field
    - 644.1.8.2. if MchTrdly=4 then break to step **644.2**
  - 644.1.9. Decrement SerialIDCount
  - 644.1.10. If SerialIDCount>=0 then go to step **644.1.3**
- 644.2. Phase 2: Determine the RDRAM's levelization timing values
  - 644.2.1. SerialIDCount=MemberMax
  - 644.2.2. Compute the 32 bit address to test the RDRAM device for levelization.
    - 644.2.2.1. SIO Register Read IOP. DEVID, SDCA=SerialIDCount, DRA=40h
    - 644.2.2.2. The DRD (MCH 090h [15:0]) now contains the RDRAM's Device ID
    - 644.2.2.3. DeviceTestAddress=MCH GBA[(DRD SHR 2)−1] SHL 23+((DRD AND 011b) * device size in bytes (from GAR[DRD SHR 2])
  - 644.2.3. Do QWORD write operation to address DeviceTestAddress with TestPattern.
  - 644.2.4. Do QWORD read operation to address DeviceTestAddress
  - 644.2.5. If data read=TestPattern then break to step **644.2.8**
  - 644.2.6. If TCDLY field of RDRAMs<Max TCDLY (from SPD)
    - 644.2.6.1. Increment the RDRAMs TCDLY registers (TDAC & TRDLY) according to the TCDLY support table.
    - 644.2.6.2. Break to step **644.2.3**
  - 644.2.7. Mark the RDRAM device to be disabled.
  - 644.2.8. Decrement SerialIDCount
  - 644.2.9. If SerialIDCount>=0 then go to step **644.2.2**

After levelization completes, one embodiment stores a number of powerdown recovery memory initialization values in the battery backed-up memory **590** of FIG. **5** as indicated in block **646**. Notably, this operation may be performed at any other stage after the appropriate values have been determined by the initialization routine. The values are saved to preserve the initialization information determined by the initialization process to this point.

When a low power state (e.g., suspend-to-RAM) is entered by the system, power to the MCH may be removed. Thus, if the initialization information is not preserved, the entire initialization process may have to be repeated. Storing key initialization information to a non-volatile memory may advantageously speed wake-up from such a low power state. The difficulty of storing such information is increased by the fact that the memory subsystem will not be functional until these values are restored.

Any non-volatile memory which can be written to may be used to store the appropriate initialization information; however, a battery backed-up memory is present in many computer systems and therefore may be a convenient choice. In one embodiment, the registers below are stored in the memory **590**.

MCH Group Architecture (GAR) registers (040–047h): These registers indicate device configuration for each group such as the number of banks and the DRAM technology (size).

MCH RDRAM Timing Register RDT (050h): This register defines the timing parameters for all devices in the channel.

MCH DRAM Control (DRAMC) register (051h): This register includes the Pool B Operation Select (PBS) bit, a memory transfer hub presence bit (MTHP), which specifies an operational mode of the MCH, and an Aperture Access Global Enable bit which prevents access to an aperture from any port before the aperture range and translation table are established.

MCH Page Policy (PGPOL) Register (052h): This register specifies paging policy attributes include a DRAM Refresh Rate (DRR) and a RMC Idle Timer (RIT). The DRR field adjusts the DRAM refresh rate and the RIT field determines the number of host bus clock cycles that the memory controller will remain in the idle state before all the open pages are closed.

MCH RPMR (053h): This register includes a Device Napdown Timer (DNT) field, an Active Devices in Pool A (ADPA) field, a Device Napdown Enable (DNE) field, and a Pool A Capacity (PAC) field. The DNT field specifies the number of host clocks the memory controller is idle before the least recently used device in Pool A is pushed out to Pool B. The ADPA field defines the maximum number of RDRAM devices in Pool A that can be in Active Read/Write or Active state at a time. The devices in Pool A that are not in Active Read/Write or Active state are in standby state. The DNE bit (when set to 1) enables the channel inactivity counter to count continuous inactivity time.

When the counter value exceeds the threshold specified by DNT, the least recently used device from Pool A is pushed to Pool B. The PAC field defines the maximum number of RDRAM devices that can reside in Pool A at a time. Devices that are not part of Pool A belong to Pool B.

MCH Group Boundary Access (GBA) registers (060–6Fh): The GBA registers contain a group ID and a value indicating the upper address limit for the group.

MCH Configuration Registers MCHCFG (0BE–BFh): These registers contain the Rambus Frequency & DRAM Data Integrity Mode fields.

Also, at this point powerdown configuration options may be programmed. In one embodiment, the self refresh and low power self refresh options are set (for each SerialIDCount: SIO Register Write IOP. INIT, SDCA=SerialIDCount, DRA=21h, DRD=400h (LSR, if SPD supports)+200h (PSR)+80h (SRP)).

Normal operation may start, as indicated in block **650**, after a few more registers are programmed for normal operation. The page policy register is set to operate normally (PGPOL RIT field (MCH 052h [2:0]) to 001b) since the page closing timer was effectively disabled for levelizing, and the power management features are enabled at this point via the RPMR register (MCH 053h). If the Pool B Select bit (MCH 051h [6]) is configured for NAP operation, a broadcast NAP entry IOP may be executed to put all devices to the NAP state. In the same I/O instruction that sets the IIO bit, set the IC bit in RICM also to one so that normal operations of the MCH may commence.

Restoring the Channel when Exiting a Low Power State

After normal operation continues for some time, the system may enter a low power state due to system inactivity or for another reason, as indicated in block **1000** of FIG. **10**. One state which the system may enter is a suspend-to-RAM (STR) state in which the MCH loses values stored in its registers. After entering the STR state, an event which causes the system to exit STR may be sensed as indicated in block **1010**. Accordingly, the BIOS powers up the MCH and other system components. The configuration registers of the MCH may be automatically reset to a default value in this process.

Accordingly, to again access memory devices on the memory channel, at least some of the configuration register values are needed. The BIOS may cause the ICH **505** to access the battery backed-up memory **590** and restore the registers listed below (saved in block **646** of FIG. **6**).

MCH GAR registers (040–047h)

MCH RDT (050h)

MCH DRAMC (051h)

MCH PGPOL (052h)

MCH RPMR (053h)

MCH GBA registers (060–6Fh)

MCH Configuration Registers MCHCFG (0BE–BFh)

After restoring values to these registers, the MCH can once again access items stored in memory when the STR state was entered, including such items as the processor context if saved. The memory devices perform self-refresh in the STR state so other data is not lost.

Next, the clock generator is started as indicated in block **1040**. The proper Rambus channel frequency is read from the MCH MCHCFG register (MCH 0BEh [11], which was restored in block **1030**). After the clock is allowed to stabilize, the MCH RAC is initialized as indicated in block **1050**. This may be accomplished by executing the MCH RAC Initialization IOP. Additionally, the DRD register may be loaded with 0000h and the MCH RAC control register load IOP executed to initialize a bus in the RAC (as discussed with respect to block **610**).

Next, current calibration is performed as indicated in block **1060**. This may be performed as discussed with respect to block **642** and FIG. **9**. In the final iteration indicated by block **940**, however, the IC bit in the RICM register may be set, allowing normal operations to immediately commence once the current calibration has completed. Thus, the resume from STR sequence may be substantially faster than the entire initialization sequence required when the system is first powered up since channel levelization, SPD querying, ID assignment, and a number of other initialization operations may be avoided.

In conclusion, a method and apparatus for initializing a memory device and a memory channel is disclosed. While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure.

What is claimed is:

1. A bus controller comprising:

a storage location including an initiate initialization operation field, wherein setting the initiate initialization operation field to a first state causes the bus controller to perform an initialization operation;

a control circuit coupled to perform an initialization operation when a value indicating the initialization operation is stored in the storage location, the initialization operation being selected from a plurality of predefined memory device initialization operations that the control circuit is capable of performing;

wherein the storage location is a control register, and wherein the bus controller further comprises a data register, and further wherein at least one of the plurality of memory device initialization operations involves data exchange between a predefined memory device control register and the data register; and

wherein the control circuit is also coupled to perform the initialization operation in response to the value being loaded into the control register, wherein the control register designates the initialization operation to be performed.

2. The bus controller of claim **1** wherein the data register and the control register are both peripheral component interconnect (PCI) configuration registers.

3. The bus controller of claim **1** wherein the initiate initialization operation field is reset to a second state when the control circuit completes the initialization operation designated by the control register.

4. The bus controller of claim **1** wherein the control register comprises:

an initialization opcode (IOP) field for storing the initialization operation;

an initiate initialization operation (IIO) field, the IIO field, when set to a first state, causing the control circuit to execute the initialization operation and then clear the IIO field to a second state;

an initialization complete (IC) field for indicating whether an initialization process has completed;

a broadcast address field which, when set, causes the control circuit to broadcast an IOP to all devices coupled to receive broadcast commands from the bus controller;

a device register address (DRA) field for specifying a register address for memory device register read and write operations; and

a serial device/channel address (SDA) field for specifying a serial device identification value for a first plurality of IOPs, a group device identification value for a second plurality of IOPs, and a bank address for a third plurality of IOPs.

5. The bus controller of claim **1** wherein the initialization operation is a memory device core initialization operation which causes the control circuit to execute, for N iterations on each of a plurality of banks:

a no operation command;

a refresh command;

a second no operation command;

a third and a fourth refresh command;

a third, a fourth, and a fifth no operation command;

a refresh precharge command;

a sixth no operation command;

a second refresh precharge command;

a first, a second, and a third calibrate command; and

a sample command.

6. The bus controller of claim **1** wherein the initialization operation is a temperature calibrate enable and then temperature calibrate operation which causes the control circuit to issue a temperature calibrate enable request packet followed by a temperature calibrate request packet to all memory devices.

7. The bus controller of claim **1** wherein the initialization operation is a current calibrate and current sample operation which causes the control circuit to send three current calibrate packets followed by one calibrate and sample packet.

8. The bus controller of claim **1** wherein the initialization operation is an initialize memory interface operation which causes the control circuit to power up a memory interface of the bus controller, to current and temperature calibrate the memory interface, and to enable a periodic current and temperature calibration of the memory interface.

9. The bus controller of claim **1** wherein the plurality of memory device initialization operations include:

a RDRAM register read command;

a RDRAM register write command;

a RDRAM set reset command;

a RDRAM clear reset command;

a RDRAM set fast clock mode command;

a RDRAM temperature calibrate enable and then temperature calibrate command;

a RDRAM Core Initialization (RCI) command;

a RDRAM Serial I/D (SIO) reset command;

a RDRAM powerdown exit command;

a RDRAM powerdown entry command;

a RDRAM current calibrate and current calibrate and sample command;

a manual current calibration of memory controller hub (MCH) Rambus ASIC Cell (RAC) command;

a load data from the data register into the MCH RAC control register command;

an initialize MCH RAC command;

a RDRAM nap entry command;

a RDRAM nap exit command;

a RDRAM refresh command; and

a RDRAM precharge command.

10. The bus controller of claim **1** wherein the control circuit further comprising an RDRAM IOP execution circuit to execute an IOP corresponding to the selected initialization operation.

11. A system comprising:

a processor;

a memory controller coupled to processor, the memory controller having a control register;

a memory bus having a plurality of memory devices coupled thereto, the memory bus being coupled to the memory controller; and

an additional memory device coupled to the memory controller, the additional memory device being accessible to the memory controller prior to initializing the plurality of memory devices, the additional memory device containing a plurality of instructions which, if executed by the system, cause the system to perform:

storing an initialization operand in the control register;

setting an initiate initialization operation field in the control register to a first state to cause the memory controller to perform an initialization operation; and

performing an initialization operation indicated by the initialization operand on at least one of the plurality of memory devices, said initialization operation being selected from a plurality of predefined initialization operations.

12. The system of claim **11** wherein the additional memory device is a non-volatile memory device and the plurality of instructions are part of a basic input/output system stored in the non-volatile memory device.

13. A system comprising:

a processor;

a memory bus having a plurality of memory devices coupled thereto;

a memory controller coupled to processor and to the memory bus, the memory controller comprising:

a first storage location including an initiate initialization operation field, wherein setting the initiate initialization operation field to a first state causes the memory controller to perform an initialization operation; and

a control circuit coupled to perform an initialization operation when a value indicating the initialization operation is stored in the first storage location, the initialization operation being selected from a plurality of memory device initialization operations that the control circuit is capable of performing; and

an additional memory device coupled to the memory controller, the additional memory device being accessible to the memory controller prior to initializing the plurality of memory devices, the additional memory device containing a plurality of instructions which, if executed by the system, cause the system to perform:

storing an initialization operand indicating the initialization operation in the first storage location;

executing the initialization operation indicated by the initialization operand on at least one of the plurality of memory devices.

14. The system of claim **13** wherein the first storage location is a control register, and wherein the memory controller further comprises a data register, and further

wherein at least one of the plurality of memory device initialization operations involves data exchange between a memory device control register and the data register.

15. The system of claim 14 wherein the first storage location is a control register, and wherein the memory controller further comprises a data register, and further wherein at least one of the plurality of memory device initialization operations involves data exchange between a memory device control register and the data register.

16. The system of claim 15 wherein the plurality of instructions stored in the additional memory device, if executed, further cause the system to perform polling of the initiate initialization operation field to determine when the initialization operation completes.

* * * * *